Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

AUTOMATIZACIÓN DE PRUEBAS FUNCIONALES DEL HLR USANDO TTCN-3

Autor: IGNACIO GARCÍA MEDINA

Directores: IGNACIO SOTO CAMPOS (UNIVERSIDAD CARLOS III) Y JESÚS GÓMEZ RUÍZ (ERICSSON)

Leganés, Enero de 2014

Título: Automatización de pruebas funcionales del HLR con TTCN-3
Autor: Ignacio García Medina
Directores: Ignacio Soto Campos (Universidad Carlos III) y Jesús Gómez Ruíz (Ericsson)

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 22 de Enero de 2014 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Como no podía ser de otra forma, mi agradecimiento más profundo es para mi familia, especialmente mi madre, mi padre y mi hermana. Ellos han sido, son y seguirán siendo lo esencial en mí y los que me han permitido llegar hasta este punto. Un agradecimiento especial debo darle también a mi prima Isabel, un gran apoyo en muchos momentos.

En la parte académica, no puedo menos que darle las gracias a Ignacio Soto por su paciencia conmigo en las idas y venidas que he tenido en este proyecto, pero siempre contando con su colaboración y consejos. También a mi manager Jesús Gómez y los anteriores, Juan Pedro Ros y Javier Sánchez-Pobre, por apoyarme desde mi empresa, Ericsson.

Después del año pasado tan horriblemente complicado que hemos sufrido en la familia, espero que esto sirva como un pequeño punto de inflexión para comenzar a mirar hacia adelante con alegría y esperanza. Os quiero.

Hay mucha gente a la que me gustaría nombrar aquí pero no podía dejar de hacerlo con todos aquellos que durante meses me preguntaban casi a diario por la evolución de este proyecto para no dejarle caer en el olvido (Vanesa, Marta, Marysia, Manu, David C. y David L. con todo mi equipo).

Resumen

La utilización de lenguajes orientados a pruebas como es TTCN-3 permite automatizar las pruebas funcionales de algunos productos. En este proyecto se han automatizado las pruebas de algunas funcionalidades básicas del HLR (Home Location Register), base de datos de la telefonía móvil.

Las funcionalidades elegidas para ser automatizadas han sido: actualización de la localización en el dominio de circuitos y en el de paquetes, cambio de IMSI y manejo del servicio suplementario de rechazo de llamadas anónimas.

La automatización de las pruebas permite ejecutarlas en cualquier momento y con coste bajo, lo cual es beneficioso para reducir los tiempos de desarrollo y para aplicar, con el mismo objetivo, técnicas de desarrollo de software ágiles en las que se implementan diferentes funcionalidades en paralelo, se va probando su correcto funcionamiento por separado y, finalmente, es necesario pasar las pruebas de todas ellas una vez todo el código esté integrado.

En este proyecto se hace una descripción del lenguaje orientado a pruebas TTCN-3, de la arquitectura de red de telefonía móvil GSM/UMTS, y del nodo al que sometemos a pruebas, el HLR. A partir de ahí se explican en detalle las funcionalildades del HLR elegidas para automatizar sus pruebas, que son: actualización de la localización en el dominio de circuitos y en el de paquetes, cambio de IMSI, y manejo del servicio suplementario de rechazo de llamadas anónimas. Finalmente se proponen casos de prueba que cubran parte de estas funcionalidades, se implementan en TTCN-3, y se evalúan sus resultados.

Palabras clave: TTCN, TTCN-3, HLR, telefonía, automatización, pruebas, Ericsson

Abstract

The usage of test oriented laguages as TTCN-3 allows the automation of functional tets of some commercial products. Within this Project several HLR (Home Location Register – mobile telephony data base) basic functionalities have been automated.

The test automation enables its execution at any time with low cost which is beneficial to reduce development time and apply agile software development techniques which implement several functionalities in parallel testing them separately but requiring all tests repetition once the code has been integrated.

This project includes a description of the test oriented language, TTCN-3, the mobile netwok architecture and the node under test, HLR.

The functionalities selected for been automated are: Update Location, Update GPRS Location, IMSI Changeover and Anonymous Call Rejection.

In this project the tested HLR functionalities are explained and the test cases are proposed and implemented in TTCN-3.

Besides, a description of tested oriented language used is done, as well as mobile network and HLR are detailed explained.

Keywords: TTCN, TTCN-3, HLR, telephony, automation, test, Ericsson

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

La telefonía móvil ha sido una de las mayores revoluciones tecnológicas mundiales de los últimos 30 años (quizá comparable a lo que ha supuesto Internet y la televisión). Los teléfonos móviles se han ido introduciendo poco a poco en la sociedad, en las casas, en la vida diaria de las personas a lo largo de todo el mundo desde que se realizó la primera llamada desde un teléfono móvil en 1973 por Martin Cooper, aunque el primer teléfono portátil comercial no llegaría hasta 1984 con la presentación del Motorola DynaTAC 8000X[i].

El ir hablando por la calle con un móvil ha pasado de parecer algo exclusivo de hombres de negocios a principios de la década de los 90, a ser lo más normal del mundo hoy en día. Algunos datos que demuestran la gran aceptación y en algunos casos la dependencia del teléfono móvil en España son que a fecha de junio de 2013, el número de líneas de telefonía móvil era de 51.927.748, frente a 46.199.064 habitantes, siendo la tasa de penetración de 112,4 líneas por cada 100 habitantes, es decir, más de un móvil con línea por persona[ii]. Si reducimos nuestro zoom y vemos cifras a nivel mundial, según el informe "Midiendo la Sociedad de la Información 2012"[iii], en el planeta ya hay más de 6.000 millones de suscripciones a móviles, frente a 7.000 millones de habitantes, y en 2014 esa cifra podría situarse en torno a los 7.300 millones de suscripciones, superando así las suscripciones a la población mundial. En cuanto a países, China se ha situado ya como el país con más suscripciones a estos servicios del mundo al superar las 1.000 millones de líneas.

De todos estos datos podemos extraer algunas conclusiones interesantes como que el teléfono móvil es algo que ya forma parte de nuestra vida, y sin el cual nos sería difícil acostumbrarnos a vivir y que ha cambiado nuestra forma de relacionarnos con la familia, amigos e incluso de conocer personas nuevas. Es la primera revolución del siglo XXI que está teniendo lugar a nivel global y a día de hoy no parece tener un final cercano.

La telefonía móvil está formada a muy alto nivel por los terminales, que son los que realmente son móviles, y por la red de telefonía móvil, compuesta a su vez por multitud de nodos que interactúan entre sí, y que es la parte fija de la telefonía móvil ya que están instalados en un emplazamiento fijo.

Los terminales móviles han evolucionado muchísimo a lo largo de estos años y lo siguen haciendo hoy en día, pasando de ser grandes aparatos con maletas donde se transportaban las baterías a casi convertidos en un pequeño ordenador personal táctil con línea de teléfono, los conocidos como "smartphones", desde cuya pantalla, gracias al acceso a Internet, tienes conexión con cualquier parte del mundo en tiempo real. Algo impensable tan sólo unos años atrás.

Las redes de telefonía móvil son muy complejas, están formadas por multitud de nodos, cada uno con una pequeña o gran funcionalidad, que tienen que entenderse entre ellos y trabajar en perfecta sintonía. Las redes son el gran desconocido de la telefonía móvil, pero a la vez su elemento crítico. Evolucionan permanentemente para añadir nuevas funcionalidades y, sin embargo, deben proporcionar alta disponibilidad: las caídas de red son extraordinariamente raras. La altísima fiabilidad de este sistema tan complejo obliga a los fabricantes a hacer pruebas funcionales, así como de Integración y Verificación muy rigurosas e intensivas de cada nuevo producto (o nueva versión del mismo) antes de poder ponerlo en el mercado. En este proyecto vamos a centrarnos en las pruebas funcionales de uno de estos nodos que componen el núcleo de la red de telefonía móvil, en concreto de su base de datos, llamado HLR (Home Location Register). Las pruebas funcionales tradicionalmente eran un proceso manual pero recientemente se ha introducido la automatización de pruebas como una técnica clave para reducir el plazo de comercialización ("time-to-market").

Ericsson, con unos 75.000 empleados en todo el mundo, es un importante fabricante de equipos de redes de telefonía móvil; en Madrid está situado uno de los centros de I+D que la empresa posee por todo el mundo y la Market Unit. En el centro de I+D se desarrollan productos clave como son el Home Data Register (HLR), Authentication Centre (AuC) y otros productos novedosos y emergentes como el SAPC/SASN para la inspección de Paquetes en tiempo real. En este proyecto se estudia la automatización de las pruebas funcionales del HLR que se ha realizado dentro de Ericsson en Madrid, incluyendo los beneficios que la automatización de pruebas pueden dar en el proceso de desarrollo de software en las grandes empresas actuales. De hecho, se puede afirmar que este proyecto es un ejemplo muy representativo de la dirección que están tomando las empresas actualmente en cuanto a procesos de desarrollo de software y pruebas.

1.2 Estado del arte

En las compañías de desarrollo de software está recibiendo mucha atención la automatización de pruebas. Se está invirtiendo muchos recursos en crear entornos útiles y flexibles para producir código de pruebas que sea reutilizable, útil, sencillo y con el mínimo mantenimiento posible.

Dependiendo del tipo de software para el que estén orientadas las pruebas hay diversas soluciones. Existen lenguajes genéricos orientados a pruebas, como el utilizado en este proyecto, pero que suelen requerir casi siempre implementaciones y/o adaptaciones *ad hoc* para el producto en concreto.

Los nuevos procesos de desarrollo de software que están siendo empleados ayudan a la automatización de pruebas ya que se basan en desarrollos iterativos e incrementales.

Existen otras herramientas de ayuda a la automatización como son los gestores de tareas, con los que podemos utilizar el código automático cuando consideremos oportuno, programando su lanzamiento (por ejemplo ejecuciones nocturnas) o hacerlo dependiente de ciertos parámetros del entorno (por ejemplo una nueva versión de código).

1.3 Objetivos

El objetivo fundamental de este proyecto es demostrar los beneficios de la automatización de las pruebas funcionales dentro del desarrollo de software en una gran empresa utilizando un lenguaje orientado a pruebas. En base a este objetivo se proponen los siguientes objetivos parciales:

- Exponer la situación de partida y la evolución del proceso de desarrollo de software
- Definir la contribución al proceso de automatización de las pruebas funcionales
- Explicar el lenguaje orientado a pruebas utilizado
- Describir el nodo bajo prueba
- Definir el entorno de pruebas utilizado
- Desarrollar las funcionalidades de las cuáles se van a automatizar sus pruebas
- Realizar el planteamiento de las pruebas
- Implementar la automatización de las pruebas
- Obtener conclusiones a la vista de los resultados

1.4 Fases del desarrollo

Para llevar a cabo este proyecto se han realizado varias fases: (1) un estudio de la situación actual de las pruebas funcionales y dónde se quiere llegar con la automatización; (2) un estudio del lenguaje elegido para la automatización y el nodo sobre el que se van a realizar las pruebas; (3) estudio de las funcionalidades elegidas e implementación de las pruebas; y, por último, (4) estudio de los resultados y conclusiones.

1.5 Medios empleados

Para realizar este proyecto se ha contado, gracias a Ericsson España, con los medios de que dispone un probador en la empresa:

- Máquina virtual con el sistema operativo SUSE Linux 10.0 para editar, compilar y ejecutar
- Acceso a la herramienta de simulación SEA
- Uso de una licencia para compilar y ejecutar de TCCN-3
- Pila EINSS7 de Tieto instalada en la máquina Linux.

1.6 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

En el capítulo segundo se presenta el nodo sobre el que se van a realizar las pruebas, el HLR, introduciéndolo como un elemento más de la red de telefonía móvil y presentando en detalle sus características y diferentes arquitecturas.

En el capítulo tercero se estudia la situación de partida de las pruebas funcionales dentro del proceso de desarrollo del software, explicando así que el cambio del proceso de desarrollo motiva el cambio hacia pruebas automáticas. También se explica la situación final que se desea tener.

En el cuarto capítulo se presenta el lenguaje utilizado para la programación de las pruebas realizadas en el proyecto, haciendo una exposición de sus características más importantes incidiendo en las que van a ser importantes durante el presente proyecto.

En el capítulo quinto aparecen las funcionalidades del HLR que se van a probar con un estudio detallado de cada una de ellas.

En el sexto capítulo se exponen las pruebas que han sido implementadas.

Y por último en el capítulo séptimo se anexan las pruebas, resultados, protocolos utilizados, etc…

Capítulo 2

Nodo Bajo Prueba: Home Location Register (HLR)

2.1 El HLR dentro de la red GSM/UMTS

Como se puede ver en la Figura 1 – Red GSM/UMTS, el HLR se comunica dentro de la red GSM/UMTS con los siguientes nodos:

- **AuC** (*Authentication Centre* – Centro de Autenticación): suele ir situado junto con el HLR, aunque pueden ser 2 nodos independientes. Cuando decimos que dos o más nodos van situados juntos, quiere decir que se instalan compartiendo hardware y sitio físico. Es el nodo encargado de generar la información necesaria (tripletas en GSM y quintetas en WCDMA) para autenticar al abonado frente a la red y a la red frente al abonado en WCDMA.
- **FNR** (*Flexible Number Register* – Registro Flexible de Números): al igual que el AuC, puede ir situado junto con el HLR. Es el nodo encargado de la Portabilidad de Números Móviles (Mobile Number Portability, MNP) y Asignación Flexible de MSISDN (Flexible Allocation of MSISDN, FAM). Las siglas MSISDN hacen referencia a Estación Móvil de la Red Digital de Servicios Integrados (RDSI) (Mobile Station Integrated Services Digital Network, MSISDN) y es el número de 15 dígitos como máximo, que identifica al abonado de la red móvil formado por CC (código del país), NDC (código de destino nacional), SN (número del

abonado). El FNR permite el encaminamiento de la llamada a través del HLR en el que están los datos del abonado.

- **MSC** (*Mobile Switching Centre* – Centro de Conmutación de Servicios Móviles): gestiona el establecimiento de las llamadas y los datos de facturación del abonado situado en una determinada área geográfica. La interconexión con otras redes hace necesaria la presencia de funcionalidades de interconexión en el MSC (IWF). Un ejemplo de esta interconexión puede ser la llamada de un móvil a un teléfono de la red fija. El MSC suele ir situado junto con el VLR.
- **VLR** (*Visitor Location Register* – Registro de Localización de Visitantes): el VLR es el nodo encargado de guardar dinámicamente cierta información de los abonados que se localizan en el área de la que el VLR está encargado. Esta información la obtiene "preguntando" al HLR. Suele ir situado junto con la MSC.
- **GMSC** (*Gateway Mobile Switching Centre* – Centro de Entrada de Conmutación de Servicios Móviles): actúa de Gateway hacia la red de telefonía fija (PSTN). Pregunta al HLR en las llamadas dirigidas hacia la red de telefonía móvil".
- **SMS-GMSC** (*Short Message Service GMSC* – Servicio de Mensaje Corto GSMC): gestiona los SMS. Pregunta al HLR en las llamadas dirigidas hacia la red de telefonía móvil.
- **SMS-IWGMSC** (*SMS InterWorking GMSC* - Servicio de Mensaje Corto Interfaz con GMSC): gestiona el envío de SMS desde la red de telefonía móvil y lo encamina al Centro de Servicio (Service Centre) adecuado.
- **SOG** (*Service Order Gateway* – Puerta de Servicio de Administración): es el interfaz entre el sistema de Administración del Operador (CAS) y la red GSM.
- **OSS** (*Operation Support System* – Sistema de Soporte de Operación): es la entidad a través de la cual el operador de la red tiene acceso a funciones de Operación y Mantenimiento.
- **SGSN** (*Serving GPRS Support Node* – Nodo de Soporte de Servicio de GPRS): GPRS permite recibir y mandar paquetes de datos a los abonados. El SGSN se encarga de la gestión de esos paquetes desde/hacia el abonado.
- **IN** (*Intelligent Network* – Red Inteligente): CAMEL provee los mecanismos necesarios para que el operador pueda proveer servicios específicos (incluso en Roaming) no contemplados en el estándar GSM/UMTS.
    - **SCP** (*Service Control Point* – Punto de Control de Servicio): aporta la lógica necesaria para que se ejecute un servicio en IN.
    - **gsmSCF** (*GSM Service Control Function* – Función de Control de Servicios de GSM): es una entidad de IN en el contexto de CAMEL.
    - **gsmSSF** (*GSM Service Switching Function* – Función de Conmutación de Servicios GSM): provee la información de las redes móviles con acceso a los servicios de IN.

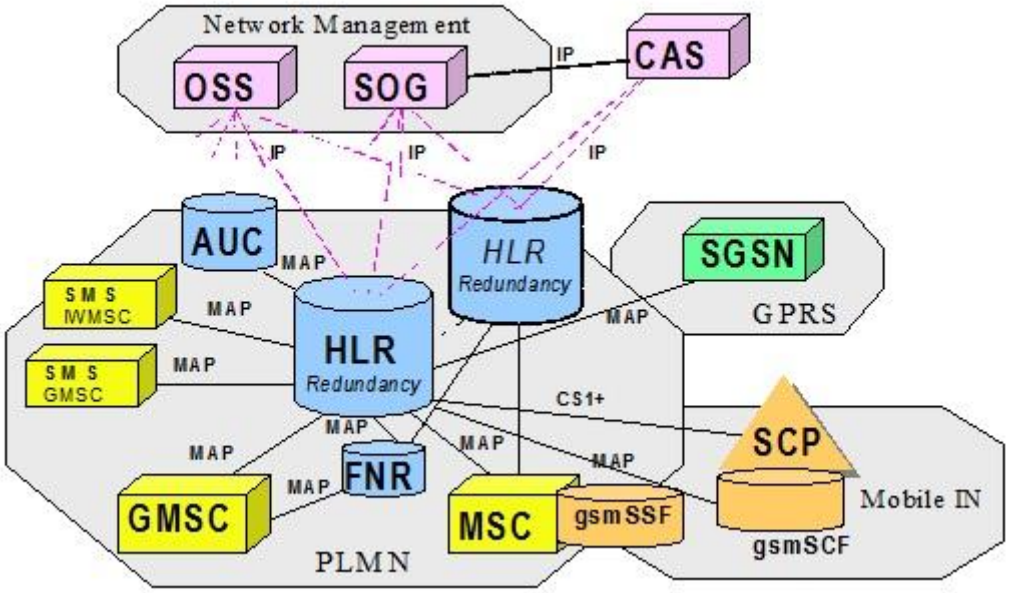


Figura 1 – Red GSM/UMTS

2.2 Arquitectura AXE

AXE es el acrónimo de "Automatic Cross-Connection Equipment" (Equipamiento de Conexión Cruzada Automática). La arquitectura AXE es una arquitectura propietaria de Ericsson cuyo desarrollo inicial tuvo lugar entre 1970 y 1977. Esta arquitectura fue el punto de inflexión gracias al cual Ericsson se hizo con una parte muy importante del mercado mundial y pasó de ser una fábrica electro-mecánica a ser un productor de alta tecnología electrónica. Los nodos más importantes que utilizan la arquitectura AXE son la MSC y el HLR.

La arquitectura AXE se basa en 3 capas: un procesador dual llamado **APZ** (AXE Control System) de procesamiento paralelo, la parte de aplicación llamada **APT** (Telephony Part of AXE) y la interfaz de entrada/salida llamada **APG** (Adjunt Processor Group) para funciones de supervisión, operación & mantenimiento y tarifación.

La familia de APZ comenzó con el APZ 210 03 y actualmente el último es el APZ 214 03. El APG es la interfaz de entrada/salida para provisionamiento. Mientras el APZ es compartido por todos los nodos, el nivel APT cambia dependiendo del nodo que se vaya a instalar: HLR, MSC, AuC, etc.

Los nodos AXE se desarrollan en un lenguaje de programación de alto nivel propietario de Ericsson llamado PLEX. La programación en PLEX es modular, se realizan bloques PLEX con funcionalidades específicas y que se comunican entre ellos utilizando señales con datos.

2.3 Funcionalidad del HLR

El Home Location Register es uno de los nodos fundamentales de la red de telefonía móvil. Sus principales funciones dentro de la red son dos: base de datos para los abonados e intermediario en la gestión de llamadas. A través del HLR no se establecen llamadas, sino que sirve de repositorio de los datos de los abonados de las redes móviles, siendo consultado por otros nodos cuando estos necesitan acceder a dichos datos.

El HLR desde el punto de vista de base de datos debe tener:

- **Primary key**: entrada única para la búsqueda de los datos. Cada abonado se identifica por medio del MSISDN-IMSI. En las diferentes operaciones que llegan al HLR, se identifica el abonado al que van dirigidas por medio del MSISD (Mobile Subscriber ISDN [Integrated Services Digital Network] Number -estación móvil de la Red Digital de Servicios Integrados [RDSI]-) o el IMSI (International Mobile Subscriber Identity -Identidad Internacional del Abonado a un Móvil-)

- **Consistencia**: los datos de abonado almacenados deben ser consistentes teniendo en cuenta la definición de servicios de acuerdo a los estándares.

- **Operaciones**: interfaces definidas para modificar y leer los datos almacenados. Estas operaciones se reciben de los diferentes nodos de la red y de Operación y Mantenimiento.

A esta funcionalidad básica se le pueden añadir más funcionalidades para telefonía y otras que aporten valor añadido para el operador.

Actualmente el HLR está evolucionando desde la arquitectura tradicional (HLR Clásico) en la que los datos de los abonados permanecían guardados en el interior del nodo a una nueva arquitectura (HLR-Server) en la que los datos se almacenan en una base de datos externa al nodo, el cual se comunica con ella para leer y modificar esos datos según convenga. Con esta nueva arquitectura los datos son mucho más accesibles para el operador. En la arquitectura Server está a punto de llegar la última revolución llamada HLR Blade Server, en la que se multiplica la capacidad del HLR.

Vamos a desgranar estas tres arquitecturas del Home Location Register.

2.4 HLR Clásico

El HLR Clásico (también conocido como HLR Monolítico) de Ericsson lleva en el mercado mundial desde el año 1992 (los primeros países en instalarlo fueron Alemania y Portugal), ha experimentado 15 Releases y está instalado en 140 países con 2,5 billones de abonados (la mitad de los abonados móviles del mundo). El centro de Desarrollo del HLR Clásico está situado en Madrid.
(http://www.ericsson.com/thecompany/company_facts/worldwide/eu/es).

El HLR Clásico de Ericsson está escrito en el lenguaje propietario de Ericsson PLEX el cual, como ya hemos mencionado, es una de las claves del éxito de muchos de los productos de Ericsson en todo el mundo. Gracias a este lenguaje se pueden introducir correcciones en código máquina en cualquier nodo para resolver un problema sin necesidad de realizar una nueva compilación software. La principal diferencia entre este HLR y las otras dos versiones (HLR Front End y HLR Front End Blade Server) es la ubicación donde se almacenan los datos. En el caso que nos ocupa, los datos de los abonados están distribuidos entre los diferente bloques software que componen el HLR Clásico.

2.4.1 Interfaces del HLR Clásico

El HLR Clásico tiene dos interfaces de entrada/salida para poder interactuar con el mundo exterior: MML y MAP como representa la siguiente figura.

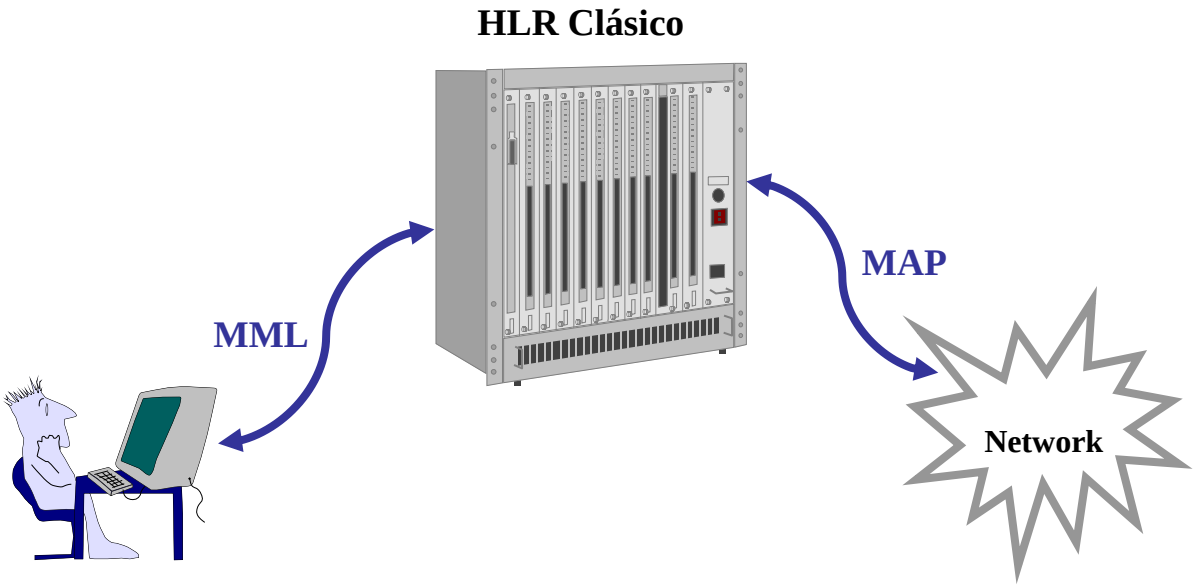


Figura 2 – Interfaces del HLR Clásico

- **Interfaz MML (Man Machine Language)**

A través del interfaz MML el HLR recibe los comandos necesarios para su configuración, como nodo perteneciente a la red, y los comandos de abonado. Estos comandos son introducidos a través de la OSS por el Operador de Red.

Actualmente existen más de 1300 comandos para el procesado de los datos de abonado. En estos comandos se puede diferenciar entre comandos de inicialización de datos (acabados en I), de borrado (acabados en E), de cambio (acabados en C) y de obtención de datos por pantalla (acabados en P).

- **Interfaz MAP (Mobile Application Part)**

La red de telefonía utiliza el Sistema de Señalización nº7 (SS7 en adelante) para la comunicación entre los diferentes nodos. El SS7 es una pila de protocolos orientados a la señalización en redes de telefonía (establecimiento de llamadas, envío de mensajes, facturación, etc…). Las aplicaciones de señalización transmiten mensajes, los cuales son transportados en paquetes. Existe un SS7 definido por ITU (International Telecommunication Union), aunque actualmente hay dos variantes que son las ampliamente utilizadas en Europa y América, estandarizadas por ETSI (European Telecommunications Standards Institute) y ANSI (American National Standards Institute), respectivamente.

Centrándonos en el caso que nos ocupa, el del HLR, tenemos la siguiente pila de protocolos de SS7 de la que hacemos uso:

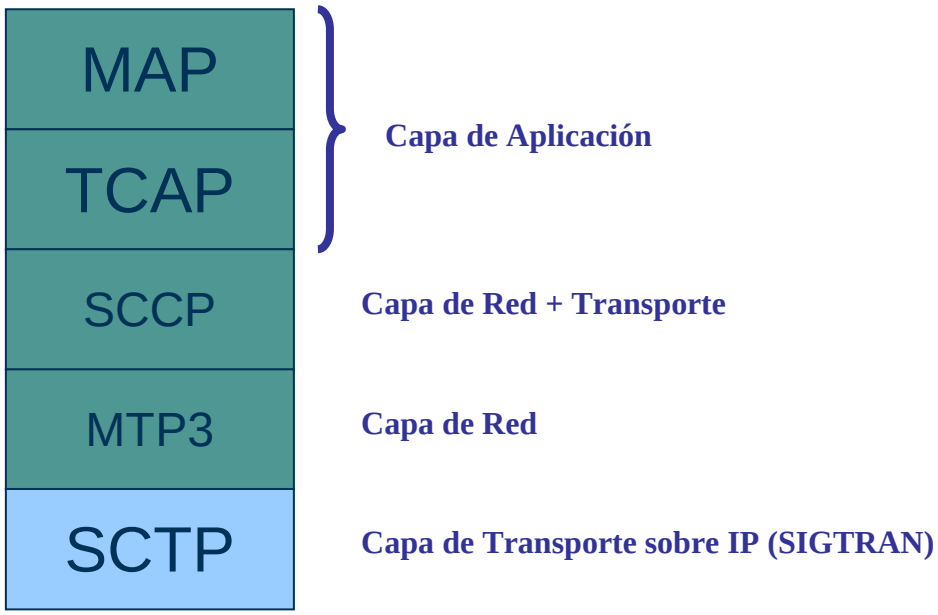


Figura 3 – Pila de Protocolos del HLR

El HLR se comunica con el resto de los nodos de la red GSM/UMTS utilizando los protocolos: MAP (en sus versiones 1, 2 y 3) y TCAP. Debajo de estas dos capas, se construye el resto de capas SCCP, MTP3.

En caso de ser necesario salir en IP, necesitamos incluir Transport Independent Signalling Connection Control Part (TI-SCCP) o SIGTRAN (SCTP). En este proyecto utilizaremos la segunda vía, la de SIGTRAN.

MAP es un protocolo SS7 que proporciona una capa de aplicación utilizada por varios nodos de la red GSM/UMTS para comunicarse entre ellos. MAP se utiliza en el HLR, VLR, MSC, EIR, AuC, SMSC y SGSN.

MAP fue creado por el grupo GSM, pero actualmente está controlado por ETSI/3GPP.

Existen dos estándares de MAP:

- MAP para GSM: 3GPP TS 09.02
- MAP para UMTS y GSM: 3GPP TS 29.002

MAP utiliza los servicios de TCAP, SCCP y MTP como se especifica en el CCITT No.7 BLUE BOOK.

MAP utiliza los servicios ofrecidos por SCCP (Signalling Connection Control Part), soportando dos versiones de SCCP:

- Signalling Connection Control Part, Signalling System no. 7 CCITT (‘Blue Book SCCP’). El HLR de Ericsson soporta este SCCP.

- Signalling Connection Control Part, Signalling System no. 7 ITU-T Recommendation (07/96) Q.711 to Q.716 (‘White Book SCCP’).

2.4.2 HLR Clásico en la red

Para integrar el HLR Clásico en una red real de un operador hace falta algún otro elemento adicional para que todo encaje y funcione adecuadamente.

Principalmente el elemento o nodo que hace falta utilizar es el APG, el cual va a dotar al operador de la interfaz de entrada/salida necesaria para provisionar el HLR. El APG permite conexiones Telnet por las cuales recibe los comandos MML que a su vez reenviará al HLR.

Otro elemento que es necesario si el tráfico MAP se quiere enviar y recibir a través de paquetes IP es un Procesador Regional (RP) con la funcionalidad de SIGTRAN. El HLR manda el tráfico a RP el cual construye el mensaje IP que se envía definitivamente a la red.

El gráfico del HLR Clásico podría quedar de la siguiente manera:

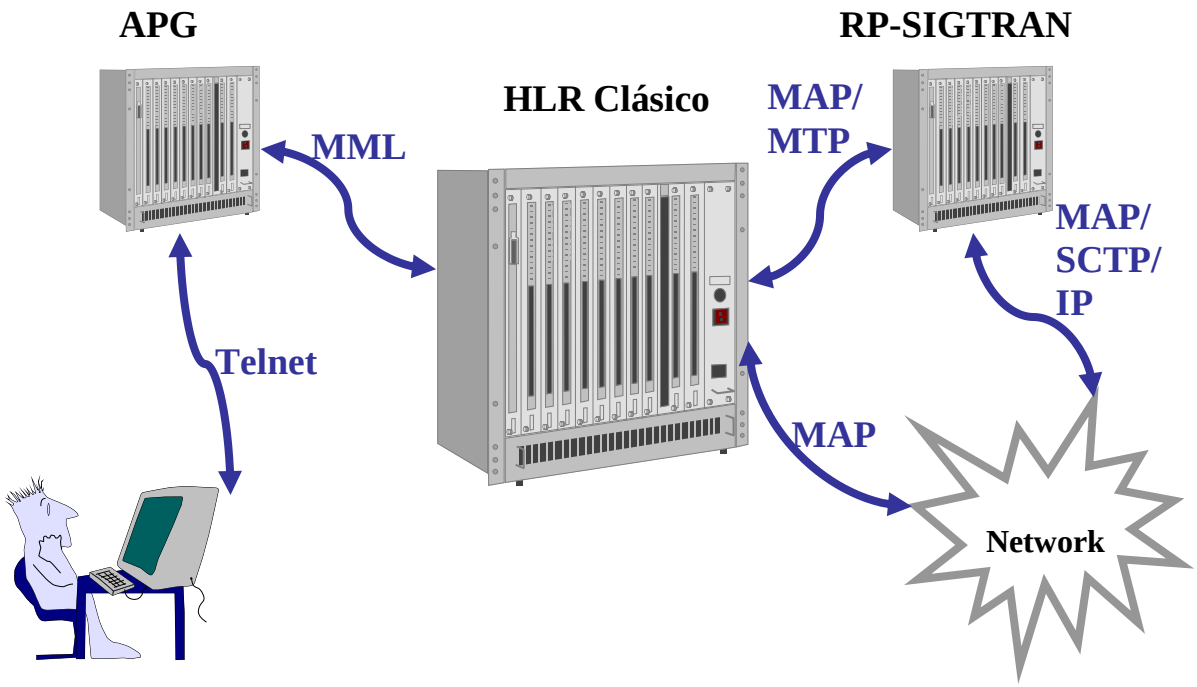


Figura 4 – HLR Clásico

2.5 HLR-FE (HLR FRONT END)

El HLR Front End (HLR-FE en adelante) nació de la necesidad de los operadores de tener los datos de abonado accesibles de una manera fácil, simple y cómoda. En el HLR Clásico la única forma que tenían de acceder a ellos es a través de interminables peticiones de salidas por pantalla realizadas sobre el nodo, el cual, a la vez que respondía a estas peticiones seguía con su tarea de atender el tráfico proveniente del resto de la red. Evidentemente las consultas de datos masivas deben hacerse en las horas valle para evitar interferir con el tráfico real.

Para solventar esta dificultad de acceso a los datos de abonado se decidió pedir a los suministradores de los Operadores un HLR en el que los datos se almacenasen en una base de datos externa sobre la que ellos podrían realizar cómodamente operaciones de búsqueda. En el caso de Ericsson, se ha decidido presentar una solución con una base de datos externa MySQL a la que se le ha añadido funcionalidad extra, una interfaz LDAP, y utilizando un nuevo "sub-nodo" para el provisionado de datos llamado "Provisioning Gateway".

Todas estas modificaciones dejan al HLR-FE con 3 interfaces: MML, MAP y LDAP.

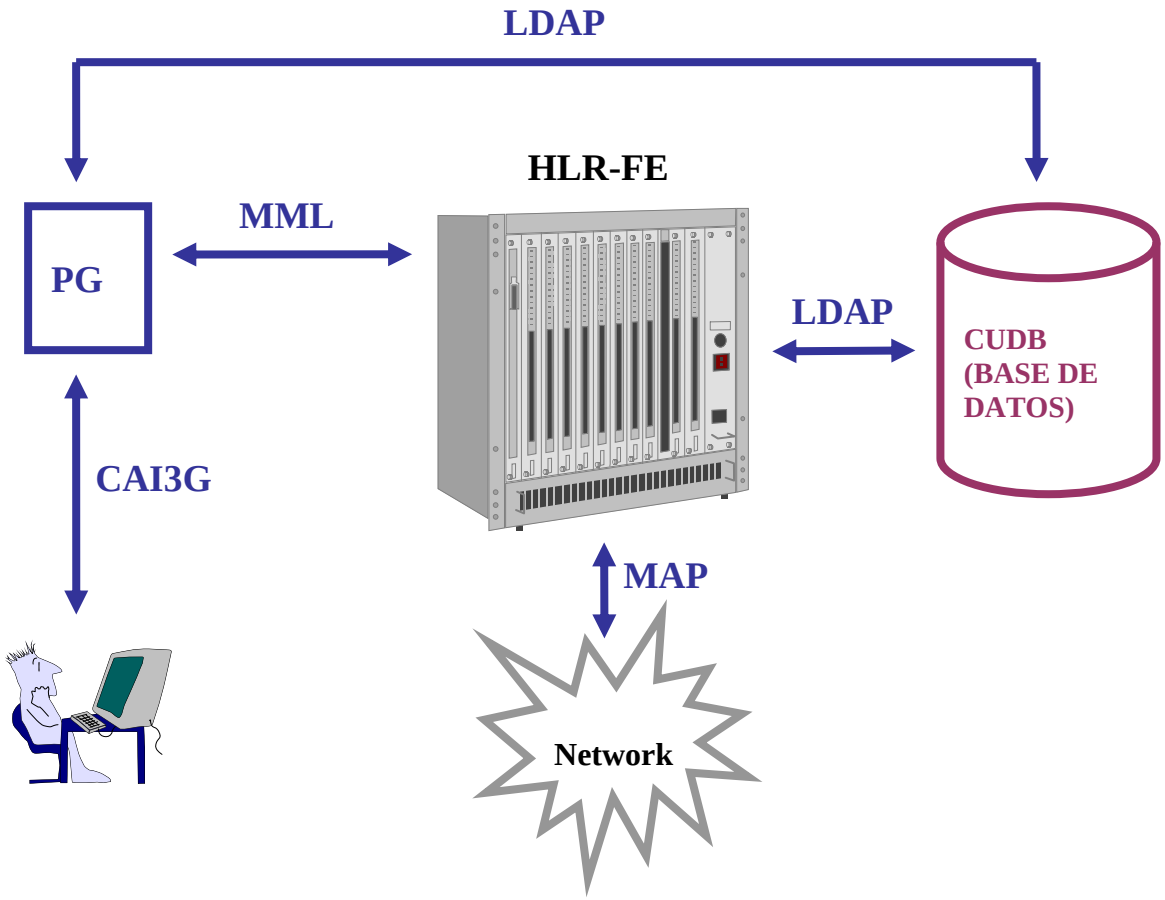


Figura 5 – Interfaces del HLR-FE

2.5.1 HLR-FE en la red

Al igual que su hermano pequeño, el HLR-FE necesita otros elementos para ser un ente completo dentro de una red de telefonía real.

El APG cumple el mismo cometido que con el HLR Clásico, es decir, reenvía los comandos MML de/hacia el HLR-FE. La diferencia es que en el Server el APG recibe comandos del PG (Provisioning Gateway), nodo que no existe en la configuración del HLR Clásico.

De la misma manera un RP cargado con la funcionalidad SIGTRAN sería necesario para enviar/recibir el tráfico MAP con IP. En otro Procesador Regional se debe cargar la funcionalidad de SDASE + LDAP para gestionar la conexión del HLR-S con la base de datos externa llamada CUDB (*Central User Data Base* – Base de Datos Central de Usuarios).

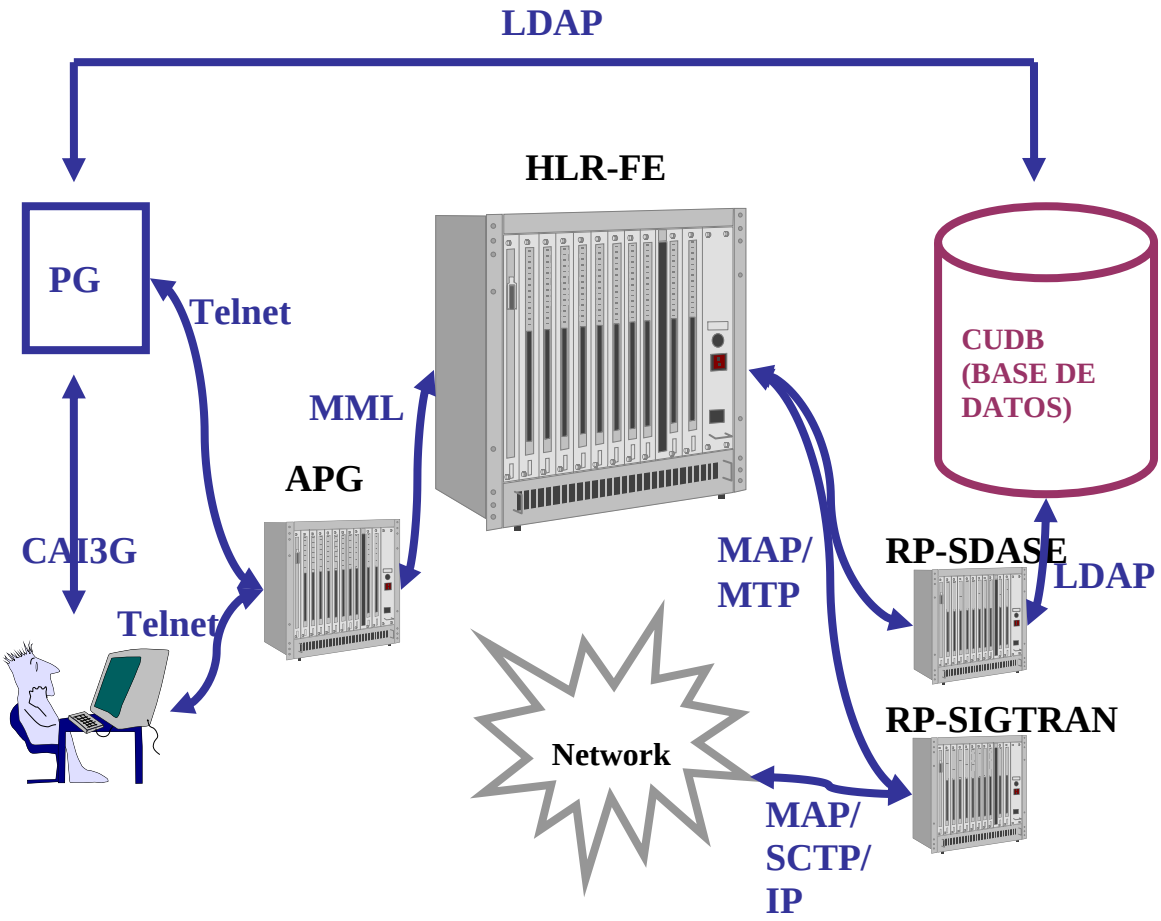


Figura 6 – HLR-FE

2.5.2 Interfaz LDAP

El protocolo LDAP (Lightweight Directory Access Protocol, Protocolo Ligero de Acceso a Directorios) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorios ordenado y distribuido. Un directorio es un conjunto de objetos con atributos organizados en una manera lógica y jerárquica. La versión actual es LDAPv3, que está especificada en una serie de Internet Engineering Task Force (IETF) Standard Track Request for Comments (RFCs) como se detalla en el documento RFC 4510.

En el caso del HLR, los atributos almacenados en la base de datos son los datos de abonado y de perfiles. Cada abonado tendrá una entrada en la base de datos con todos sus atributos y/o perfiles asociados.

El cliente LDAP puede realizar las siguientes operaciones:

- **Start TLS**
- **Bind** → Autenticarse y especificar una versión del protocolo LDAP

- **Search** → Buscar y/o obtener entradas de directorio
- **Compare** → probar si una entrada nombrada contiene un valor de atributo dado
- **Add** → Añadir una nueva entrada
- **Delete** → Borrar una entrada
- **Modify** → Modificar una entrada
- **Modify Distinguished Name** (DN) → Modificar o renombrar una entrada
- **Abandon** → Abortar una petición previa
- **Extended Operation** → Operación genérica usada para definir otras operaciones
- **Unbind** → Cerrar la conexión

El PG, usando LDAP, manda los comandos al HLR y recibe una salida por pantalla con los atributos que debe cambiar y su nuevo valor; partiendo de esta salida por pantalla el PG modifica la Base de Datos mediante operaciones LDAP.

El HLR realiza búsquedas de los atributos de los abonados sobre los que se tiene que realizar alguna operación, de administración o de tráfico. También puede realizar modificaciones de estos atributos de abonado originadas por operaciones de tráfico únicamente, ya que los cambios por administración los realiza el PG exclusivamente.

2.6 HLR-FE Blade Server

La siguiente evolución del HLR que está en muy pocas redes reales de telefonía es el HLR-FE Blade Server.

Es bastante simple de entender la evolución que supone esta arquitectura que explicada en pocas palabras es: un HLR compuesto por varios HLR-FE funcionando en paralelo, con lo cual teóricamente el nodo multiplica sus prestaciones. El número mínimo de HLR en paralelo es 2 y el máximo 19 en la solución que Ericsson desarrolla.

Desde la red se seguiría viendo como un único HLR, aunque realmente tuviera varios internamente actuando. En la figura 9 podemos ver esta idea en un dibujo.

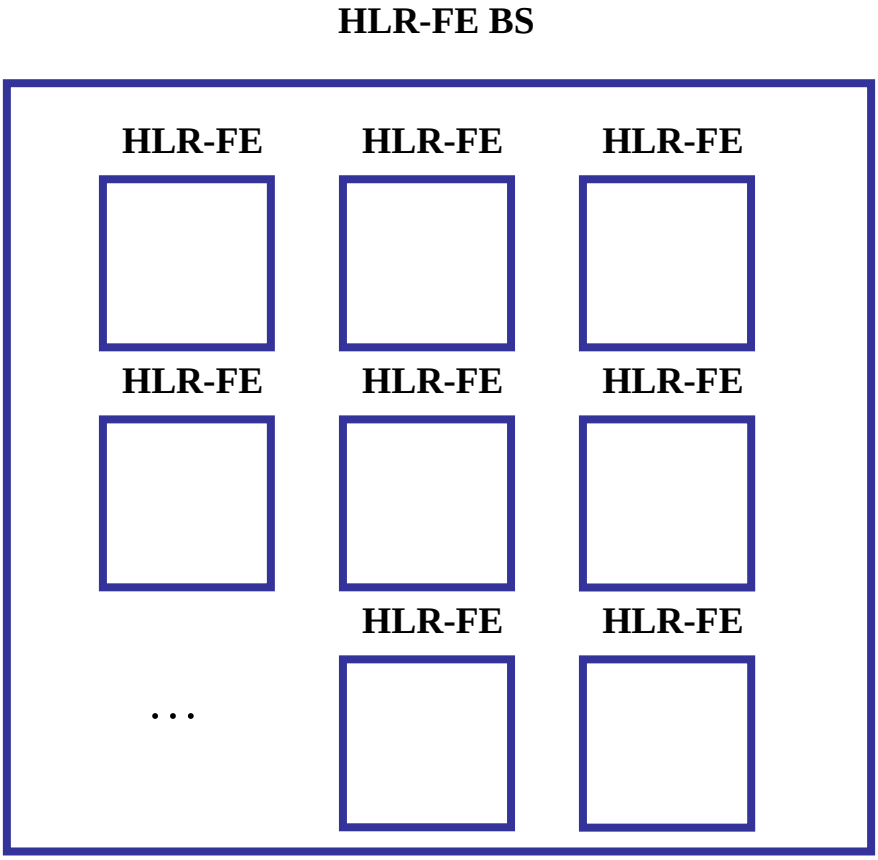


Figura 7 – HLR-FE Blade Server

La otra novedad del HLR-FE Blade Server, en el caso de la solución de Ericsson, es que la funcionalidad que en el HLR Clásico o FE estaba en los Procesadores Regionales (RP) ahora se migra para que se ejecute sobre Linux y poder integrarla en el nuevo Hardware del HLR-FE Blade Server.

2.7 Conclusiones

Uno de los atractivos de este proyecto es que se han realizado pruebas a un nodo real y clave dentro de la red de telefonía móvil. Como hemos visto en este tipo de nodo hay almacenados unos 2.500 millones de abonados en diferentes redes por lo que es un producto muy sensible y en el que cualquier fallo puede ser catastrófico y suponer la pérdida de conexión de muchísimas personas.

Acabamos de ver en detalle las tres arquitecturas del HLR en el mercado actualmente, siendo el HLR Clásico el que está instalado en el 90% de las redes. Tanto HLR-FE como HLR-FE Blade Server están empezando a ser instalados en algunos operadores recientemente.

Capítulo 3

Procesos Tradicionales de Pruebas Funcionales del HLR

En este capítulo vamos a analizar la situación de partida del proyecto. El marco es el departamento de desarrollo del HLR en el cuál vamos a analizar primero el proceso que se estaba utilizando y las necesidades que se identificaron en cuanto a pruebas cuando se evolucionó hacia un nuevo proceso más eficiente en el desarrollo del software que dejó de ser lineal pasando a ser desarrollo paralelo de varias funcionalidades.

3.1 Motivaciones para el cambio

Por encima de todo ha habido un motivo fundamental para que se tomase la decisión de automatizar las pruebas funcionales del HLR: un cambio en el Proceso de Desarrollo del producto.

Durante los últimos años el proceso para el desarrollo del software con las nuevas funcionalidades requeridas por los operadores en el HLR era un proceso lineal (Figura 8). Durante este proceso se agrupaba a los diseñadores en equipos para diseñar e implementar (pasos 1 y 2) las diferentes nuevas funcionalidades. En las pasos 3 y 4 eran los probadores los que entraban en escena para realizar tanto pruebas funcionales como de Integración y Verificación.

1) Diseño
2) Implementación
3) Pruebas Funcionales
4) Pruebas de Integración y Verificación

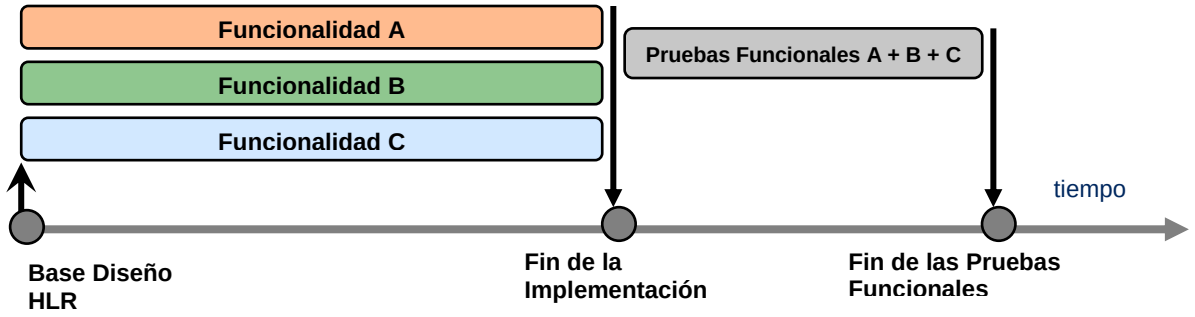


Figura 8 – Proceso de Desarrollo Lineal

Pero debido a la presión del Mercado mundial para llegar antes con una nueva versión del producto o Release (conocido como "time to market") y al mismo tiempo reducir costes, se ha decidido cambiar hacia un Proceso que podemos llamar "incremental", basado en la metodología Agile/Scrum[iv].

En este nuevo proceso, existen unidades de desarrollo software cuasi-autónomas a las que se les entrega una entrada en forma de requisitos y de las que se obtiene el código implementado y con las pruebas funcionales ya realizadas. Estás unidades o equipos trabajan en paralelo unas con otras desarrollando nuevas funcionalidades distintas entre sí y entregando el nuevo software en momentos distintos del tiempo. Este desarrollo en paralelo implica un problema añadido: cuando un equipo va a entregar, su base de diseño (código que el equipo tomó como origen para introducir sus cambios) puede haber cambiado por lo que se necesita unir el código realizado y hacer una repetición de las pruebas de las funcionalidades entregadas desde su base de diseño, es decir, volver a ejecutar esas pruebas, para asegurar que todo sigue funcionando perfectamente.

¿Qué implicaciones tiene esta repetición de las pruebas? Implica que las pruebas funcionales de cada nueva funcionalidad tendrán que ser repetidas varias veces a lo largo de un proyecto, por lo tanto, es necesario automatizar las pruebas. El beneficio de automatizar las pruebas está en que sean repetidas varias veces de manera sencilla, y con este nuevo proceso lo encontramos claramente.

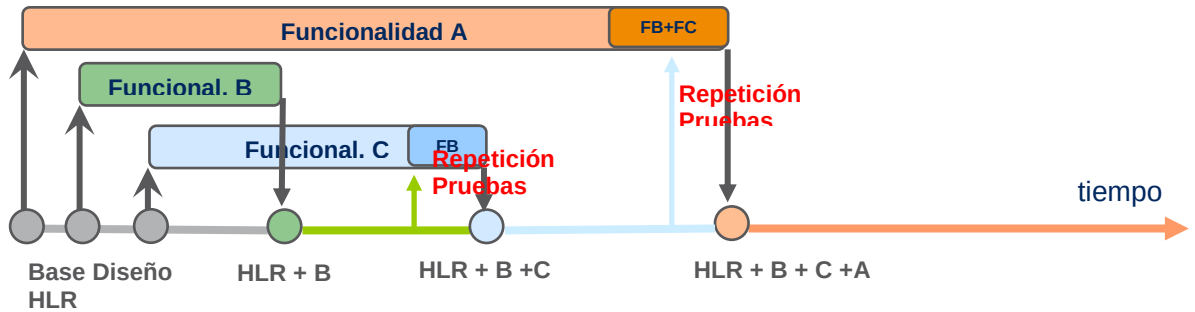


Figura 9 – Proceso de Desarrollo Ágil

3.2 Situación de Partida

La situación de partida de las pruebas del HLR era que todas y cada una de ellas se realizaban de forma manual. Existían numerosos documentos donde se agrupaban las pruebas por funcionalidades, en los que cada prueba estaba dividida en 3 partes:

- **Preparación o precondiciones**: en todos y cada uno de los casos de pruebas funcionales se debe partir de la misma situación inicial del nodo bajo prueba, podríamos llamarlo estado neutro. Durante la preparación de la prueba se crean las condiciones que van a ser necesarias para su realización, como creación del abonado, proveerle de ciertos servicios, localizarle, etc...

- **Prueba**: una vez puesto el nodo en las condiciones requeridas para la prueba, se realiza la prueba en sí misma.

- **Restauración**: una vez ejecutada la prueba, se deja al nodo en las mismas condiciones que tenía al comienzo, es decir, en el estado que hemos llamado neutro.

A su vez, cada uno de estos documentos, los cuáles se conocen como "Test Description" (TD) que agrupan las pruebas de una determinada funcionalidad, podían tener unas preparaciones generales comunes para todos los casos de prueba y unas condiciones de restauración también generales.

La situación, para el HLR, era que se tenían unos 20.000 casos de prueba agrupados en unas 90 TDs. Para ejecutar uno de estos casos de prueba se tenía que hacer manualmente.

Las pruebas del HLR estaban compuestas de:

- **Administración**: existen comandos de abonados y de configuración del nodo. El HLR acepta comandos de creación y borrado de abonados, así como comandos para modificar los datos y perfiles de dichos abonados. También acepta comandos que no tienen que ver con los abonados, simplemente son para la configuración del propio nodo dentro de la red.

- **Tráfico**: recepción/envío de tráfico desde/hacia el resto de la red. El HLR tiene una interfaz para el envío y recepción de tráfico MAP con el resto de la red, y es por el único que se puede comunicar con el resto de los nodos de la red de telefonía.

En las pruebas de las que partimos, la parte administrativa se ejecutaba de forma manual, introduciendo el comando por consola y analizando su salida por pantalla de respuesta. Pero para el tráfico, se necesitaba utilizar un generador de tráfico: MGTS de la empresa Catapult (IXIA). Vamos a describir con detalle el entorno virtual en el que se desarrollaban las pruebas manuales.

3.3 Entorno manual de pruebas de funcionalidad

3.3.1 Administración

Los comandos de administración son comandos MML (Man-Machine-Language, Lenguaje Hombre-Máquina) y se pueden enviar desde diferentes aplicaciones. La forma más común en el entorno manual era desde una herramienta llamada Winfiol que se ejecuta sobre Windows. Con esta herramienta el probador se conecta al nodo y es capaz de enviar comandos MML y recibir la salida por pantalla de respuesta del nodo. Winfiol (Figura3) tiene otras funciones de ayuda como que permite abrir un fichero de texto y enviar los comandos desde él, uno por uno o agrupando varios.

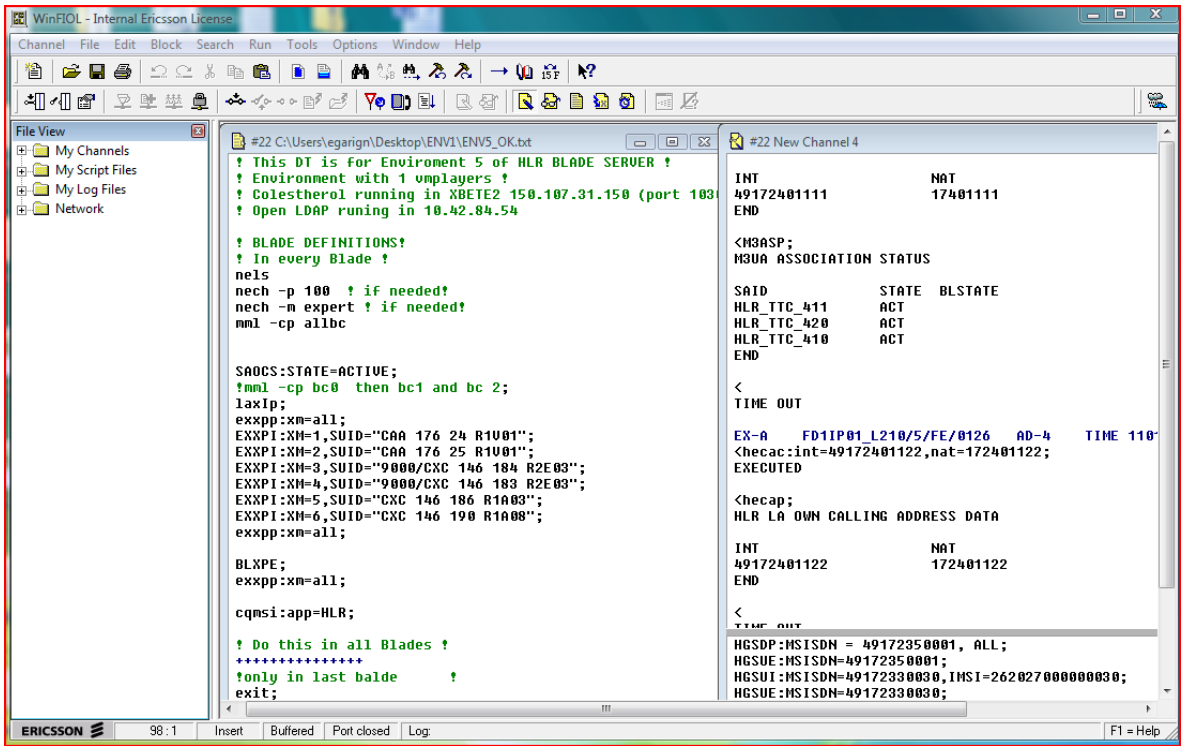


Figura 10 – Winfiol

3.3.2 Generador de Tráfico

El MGTS (Figura 11) es una herramienta de generación de tráfico muy potente que permite crear un diagrama de estados en los que el probador puede editar los mensajes que desee mandar y chequear los datos que necesite en los mensajes que recibe. También el probador puede decidir a qué estado se pasa dependiendo de si ese chequeo ha sido positivo o negativo. Este es un ejemplo (Figura 11) de una localización de área en el que podemos observar:

Estado inicial: START (blanco)
Envío Mensaje Update Location (rosa)
Múltiple recepción de Insert Subscriber Data dependiendo de la longitud (naranjas)
Envío del Result del Insert Subscriber Data (amarillo)
Recepción de otro Insert Subscriber Data (bucle)
Recepción del Result del Update Location (morado)
PASS (verde)

Temporizador de 15 segundos si no recibe nada (Rojo)
FAIL (Rojo)

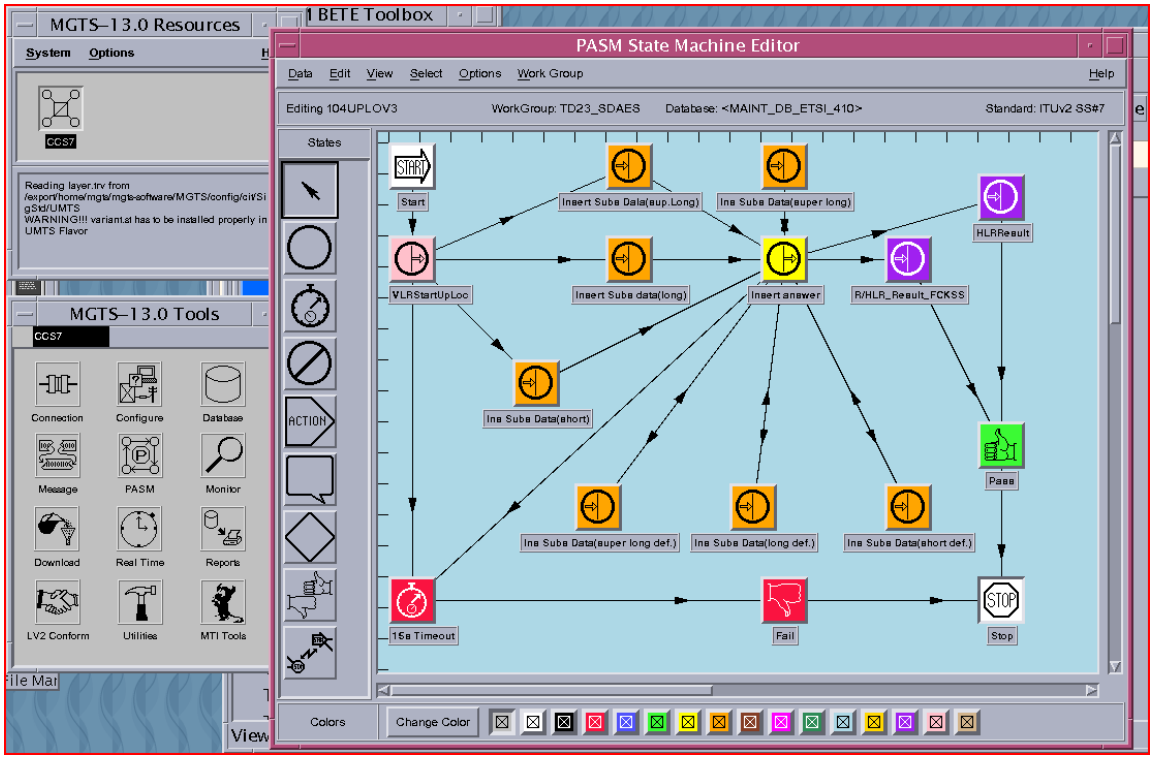


Figura 11 – MGTS

El editor de mensaje (Figura12) permite importar los valores de una base de datos interna o poner el valor en cada campo. En esta captura podemos ver el mensaje Update Location (azul) (Begin/Invoke) con el valor del IMSI.

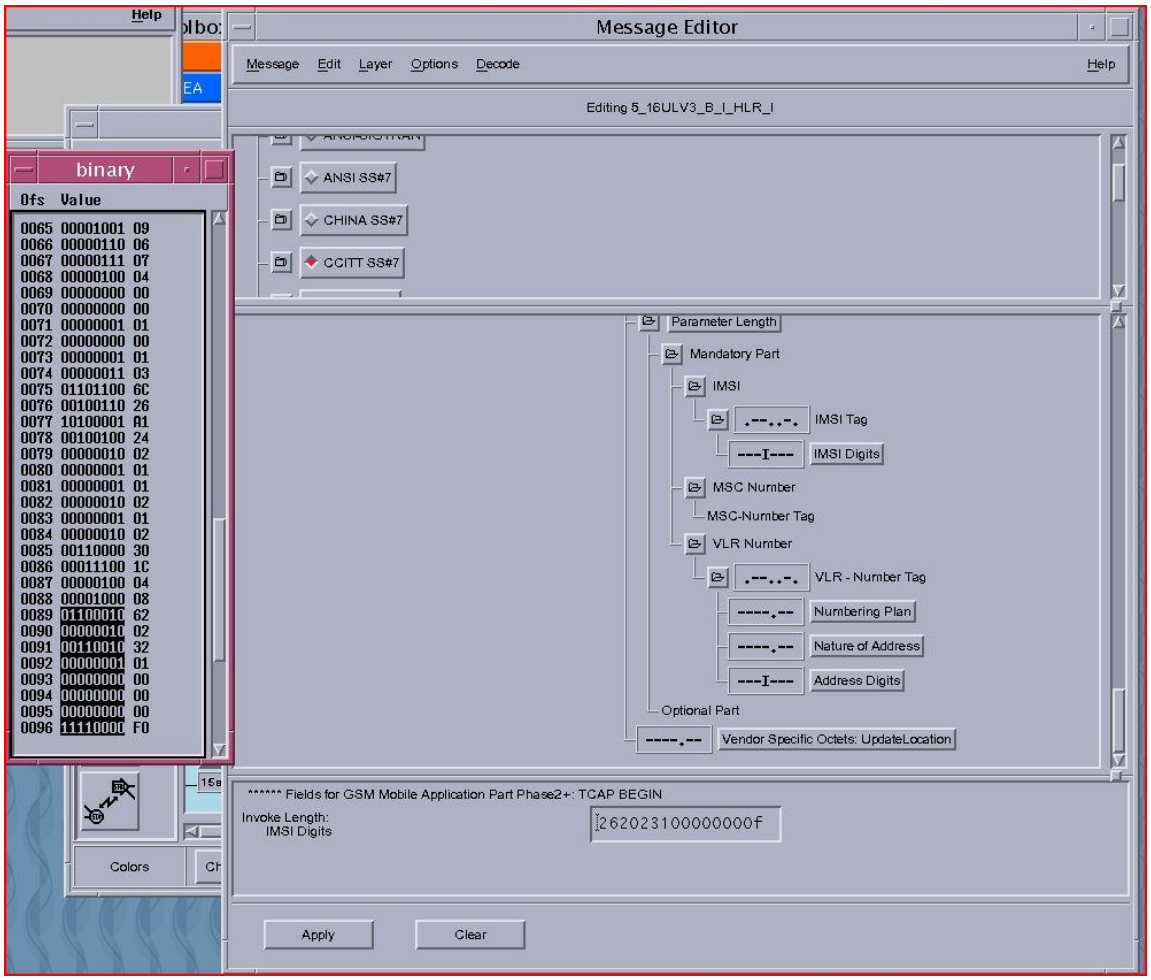


Figura 12 – Editor de mensajes de MGTS

3.3.3 Aplicación de simulación del Software del HLR

Ericsson utiliza un software desarrollado por TATA Consultancy llamado SEA (Figura6), el cual permite cargar el mismo binario que se carga en las maquetas reales, comportándose exactamente igual en cuanto a funcionalidad se refiere (no a prestaciones). Esto permite realizar las pruebas funcionales evitando el coste y las limitaciones de utilizar las maquetas reales. Evidentemente, para pruebas de Integración y Verificación se ha de utilizar hardware real.

Durante este proyecto hemos utilizado esta herramienta llamada SEA, en la cual cargamos el software del HLR para poder conectarnos a él y realizar las pruebas pertinentes.

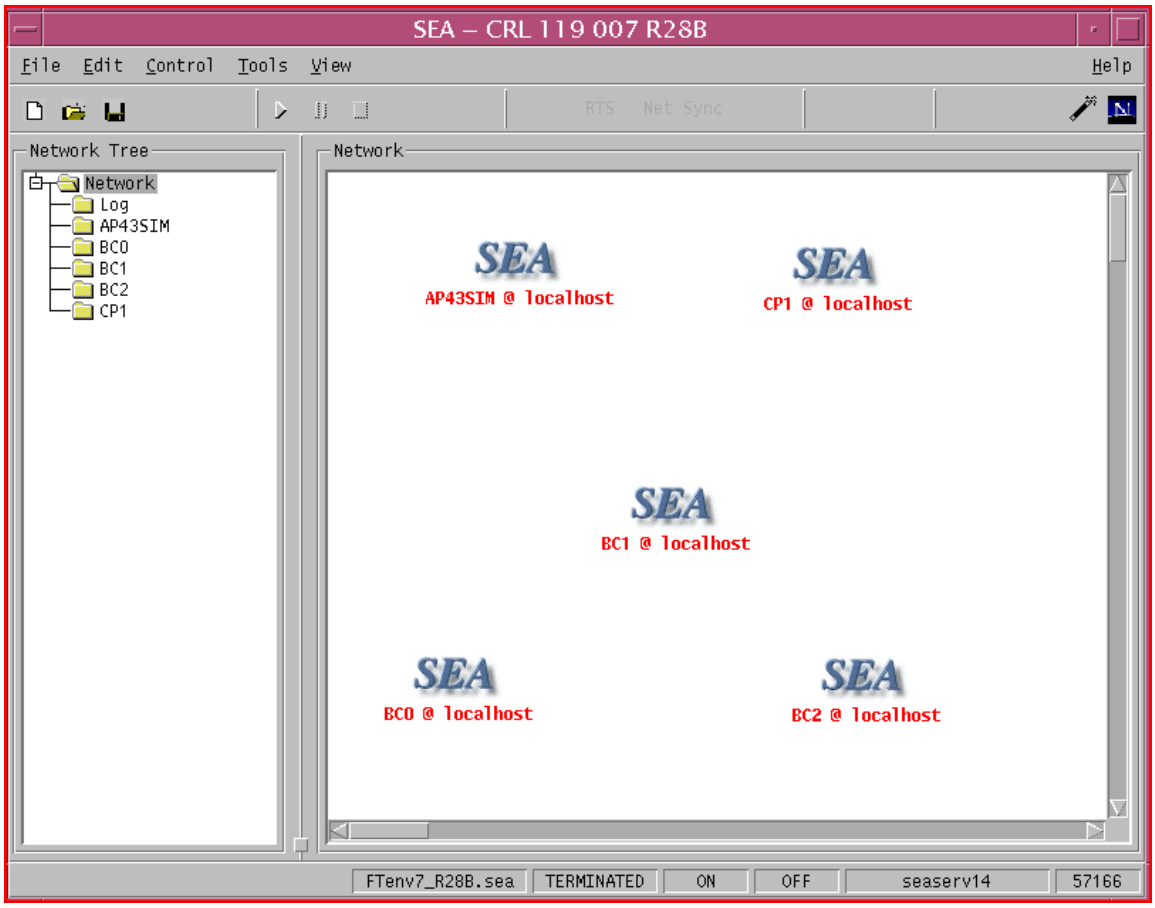


Figura 13 – SEA

3.4 Entorno automático de pruebas de funcionalidad

3.4.1 Aplicación de simulación del Software del HLR

En el entorno de pruebas automáticas vamos a utilizar el mismo software que en el entorno manual, la herramienta SEA.

3.4.2 Generador de tráfico y comandos

En el entorno automático vamos a programar todas las pruebas automáticas y compilarlas. Dicha compilación generará un ejecutable que va a ser el que lance y reciba tanto comandos como tráfico contra el HLR levantado en la herramienta SEA.

La compilación ha sido realizada en un Sistema Operativo Linux. En este caso hemos utilizado Linux SUSE10 para hacerlo. Por motivos de eficiencia, el Linux estaba

embebido en una máquina virtual, con lo que hemos aprovechado mucho mejor el hardware disponible pues se podía usar la misma máquina simultáneamente para otras funciones.

3.5 Pila EINSS7

En la parte de tráfico, el programa el TTCN-3 hará uso de una pila número 7 de Tieto instalada en la máquina Linux, la cual completa la pila hasta salir por IP y ha sido configurada con los siguientes puntos de señalización que por supuesto deberán ser utilizados a la hora de programar las pruebas:

- Punto de señalización del HLR = 300
- Número de Subsistema del HLR = 6
- Puntos de señalización de los nodos simulados por TTCN-3 = 410, 411 y 420
- Ruta (GT) para el HLR
    - Number Series = 4917240*
    - Numbering Plan = E163/E164 (ISDN)
    - Nature of Address = International Number
    - Tranlation Type = 0
- Diversas Rutas para todos los posibles nodos simulables

El programa TTCN-3 hará uso de los puertos TCAP para hacer el “bind” a la pila y mandar el mensaje. Los mismos puertos recibirán el mensaje de vuelta y lo procesarán según hayamos programado en la pila.

3.6 Conclusiones

Como resumen, partimos de un desarrollo del software utilizando un proceso lineal con unas pruebas de funcionalidad ejecutadas manualmente utilizando herramientas como Winfiol y MGTS. La ambición, dentro de la que se enmarca el trabajo de este proyecto, era automatizar estas pruebas para poder migrar a un proceso ágil en el que es necesario hacer cierta repetición de las pruebas.

Capítulo 4

TTCN-3

TTCN-3[v] (Testing and Test Control Notation Version 3) es el lenguaje con el que se van a realizar las pruebas automatizadas en este proyecto. TTCN2 es el predecesor del, ampliamente utilizado, TTCN-3. Este lenguaje está desarrollado y mantenido por ETSI.

TTCN-3 se ha utilizado durante más de 15 años en pruebas y certificaciones en la industria, especialmente en la de Telecomunicaciones, más concretamente en grandes empresas como Nokia, Motorola, Huawei, Ericsson (más de 2000 licencias de TTCN-3).

TTCN-3 tiene características que lo hacen muy atractivo y flexible:

- No está ligado a ninguna herramienta específica.
- No está ligado a ningún entorno de compilación/ejecución concreto, ni siquiera a un sistema operativo.
- No depende del nodo que está bajo prueba. Únicamente hay que tener conexión con él para poder mandar y recibir mensajes.

Para poder ejecutar casos de prueba de TTCN-3, necesitamos un "Sistema TTCN-3". Este sistema podemos dividirlo en 3 partes:

- **TCI** - Interfaz de Control, la cual nos permite seleccionar qué casos lanzar, cuándo y contra qué sistema bajo prueba (SUT, System Under Test), obteniendo unos resultados en ficheros de registro (log's).
- **TTCN-3 ejecutable** - es la compilación del caso de prueba en sí mismo.

- **TRI** - Interfaz en tiempo de ejecución, son las conexiones hacia el SUT y la adaptación a la plataforma donde estemos lanzando las pruebas de TTCN-3.

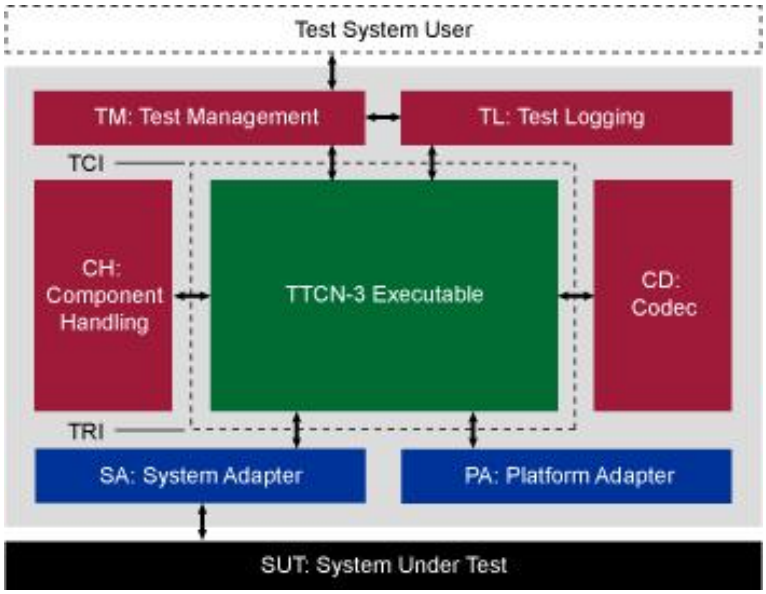


Figura 14 – Sistema TTCN-3 (tomada de [vi])

Actualmente existen cuatro herramientas comercializadas para TTCN-3 en el mercado, y se está discutiendo la creación de una herramienta de código abierto.

Otras de las virtudes de TTCN-3 son:

- **Escalabilidad**: permite añadir cambios fácilmente (ej. cambio de componentes) sobre pruebas ya implementadas.
- **Extensibilidad**: permite mapear tipos de datos de manera muy simple, actualmente admite ASN.1, XML y C. En la Tabla1 podemos ver el mapeo de datos de ASN.1 a TTCN-3:

ASN.1 TYPE	TTCN-3 TYPE	*ASN.1 EXAMPLE*	*TTCN-3 EQUIVALENT*

BOOLEAN	boolean	Message ::= SEQUENCE {
		version [0] IMPLICIT INTEGER(0..99),
		mId [1] MId,
		messageBody CHOICE {
		messageError [2] ErrorDescriptor,
		transactions [3] SEQUENCE OF Transaction
		}}
			type record *Message* {
			integer *version* (0..99),
			MId mId,
			MessageUnion messageBody
			}
			
			type union *MessageUnion* {
			ErrorDescriptor
			messageError,
			record of *Transaction*
			transactions
			}

INTEGER	integer		
REAL	float		
OBJECT IDENTIFIER	objid		
BIT STRING	bitstring		
OCTET STRING	octetstring		
SEQUENCE	record		
SEQUENCE OF	record of		
SET	set		
SET OF	set of		
ENUMERATED	enumerated		
CHOICE	union		
NULL	type enumerated <identifier> { NULL }	MyType ::= NULL	type enumerated *MyType* { NULL }

Tabla 1 – Tabla equivalencia ASN1 TTCN-3 (tomada de [vii])

- **Repetir pruebas**: permite una fácil y cómoda repetición de las pruebas.

- TTCN-3 ha sido utilizado para realizar pruebas en diferentes sectores como automoción, finanzas, banca y servicios médicos entre otros.

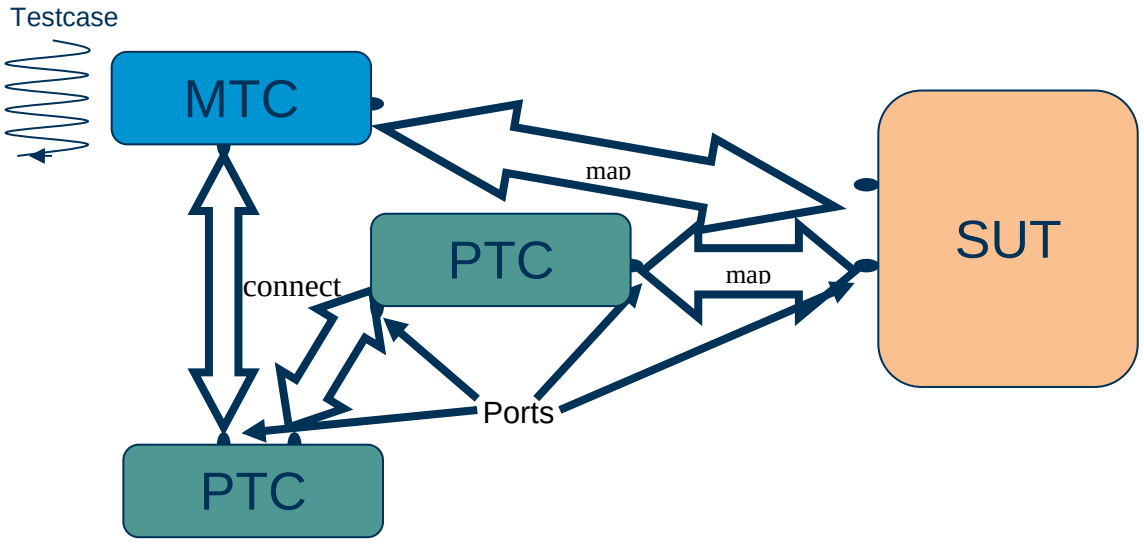
TTCN-3 es un lenguaje totalmente orientado a pruebas y diseñado para ser utilizado en entornos muy complejos, como puede ser la red 2G/3G. Hay ciertas características del lenguaje que le hacen especialmente atractivo para las pruebas:

- Concepto de veredicto, es decir, ir estableciendo en tiempo de ejecución si la prueba va pasando o en qué punto ha fallado.
- Potente mecanismo de comparación entre respuesta recibida y esperada.
- Preservación del orden de llegada de los eventos externos, los cuales se pueden comparar con multitud de alternativas.
- Posibilita la ejecución de pruebas concurrentes contra el SUT.
- Configuraciones dinámicas de las pruebas.
- Permite parámetros de configuración, lo que hace fácilmente adaptables los casos de prueba a multitud de entornos.
- Temporizadores.

4.1 Modelo de funcionamiento de las pruebas en TTCN-3

Lo primero que hay que entender cuando trabajamos realizando pruebas en TTCN-3 es cómo este lenguaje estructura los casos de pruebas. Cada caso de prueba comienza con la palabra clave *testcase* y cuando este arranca se crea un proceso llamado Componente Principal de Prueba [Main Test Component (MTC)]. Este proceso MTC se puede considerar el proceso principal del caso de prueba. El MTC puede crear uno o más procesos llamados Componentes Paralelos de Prueba [Parallel Test Component (PTC)], que se pueden asemejar a procesos hijos. Todo los procesos pueden comunicarse entre sí en caso de ser necesario, pero introduce más dificultad al caso de prueba (en este proyecto no hemos considerado necesaria esa comunicación). Tanto el MTC como los PTC's pueden conectar sus puertos contra el SUT y mandar y recibir mensajes. Esta operación de conectar los puertos consiste en iniciar un diálogo con el SUT a través del interfaz definido para cada caso, por ejemplo los puertos TCAP realizan un bind a la pila de protocolo número 7 al ser conectados.

En el siguiente gráfico se refleja este modelo de trabajo:



MTC: Componente de Prueba Principal (Main Test Component)
PTC: Componente de Prueba Paralelo (Parallel test Component)

Figura 15 – Configuración de procesos en TTCN-3

4.2 Estructura interna del lenguaje TTCN-3

La escritura con TTCN-3 se estructura en dos tipos de ficheros principalmente:

- Ficheros con extensión **.ttcn**, dentro de los cuales el código se estructura en Módulos. Un Módulo puede importar definiciones de otros Módulos, y puede ser una librería, una colección de casos de prueba, una colección de plantillas, de funciones, etc...

- Ficheros con extensión **.cfg**, dentro de los cuales se incluyen:
  - Parámetros de los Puertos
  - Parámetros de entrada del binario, su utilidad radica en que no hay que recompilar el binario para poder cambiar el valor de estos parámetros entre una y otra ejecución

En la Figura 16 podemos ver que dentro de un Módulo puede haber una parte de Definiciones y otra de control. Y fuera del Módulo podemos ubicar los parámetros de entrada del fichero de configuración

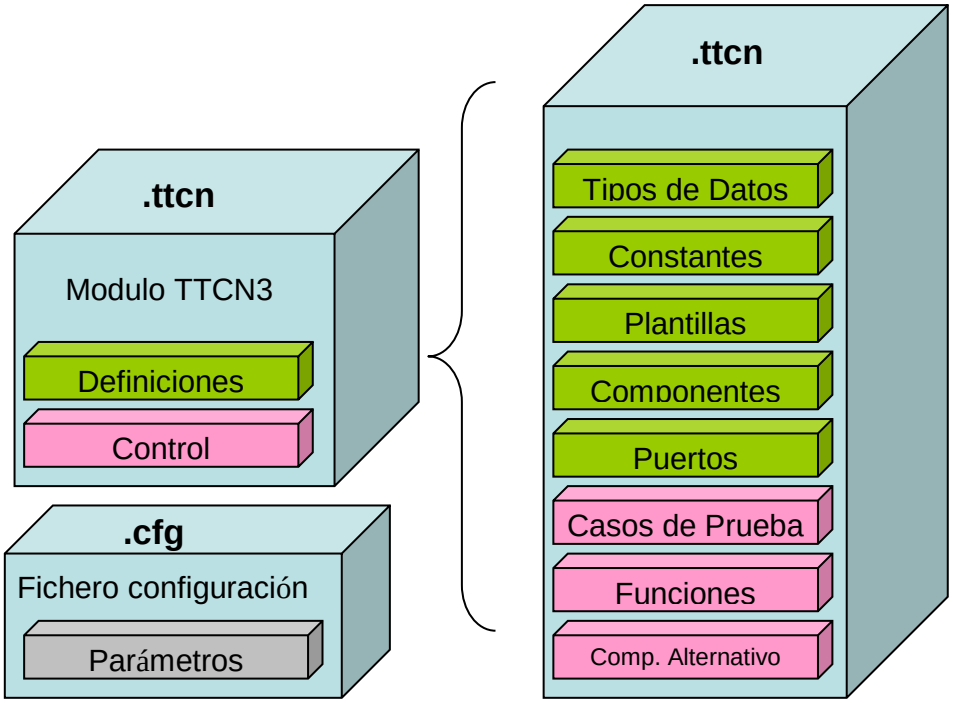


Figura 16 – Módulos TTCN-3 (tomada de[viii])

Dentro de las Definiciones podemos incluir estructuras tan diversas en un lenguaje de programación como son los Tipos de Datos, Constantes, Plantillas, Componentes y Puertos. Vamos explicar cada una por separado para saber cómo se utiliza en TTCN-3.

4.2.1 Tipos de datos

- Tipos básicos

  Boolean, Integer, Float, Char, Universal Char and Verdicttype

- Tipos String

  Bitstring, Hexstring, Octetstring, Charstring, and Universal Charstring

- Tipos estructurados

  Record (ordered structure), Record Of (ordered list), Set (unordered structure), Set Of (unordered list), Enumeration, Union and Anytype.

- Tipos de Configuración

Los Tipos de Configuración son los tipos más identificativos del TTCN-3 y alejados de otros lenguajes de programación. Se componen de:

- **Puertos (Port)**: los puertos se definen como una lista de mensajes/procedimientos que se pueden recibir o enviar a través de ellos.
- **Componentes (Component**): los Componentes son el conjunto de puertos sobre el que se van a hacer las pruebas. También pueden incluirse variables locales, constantes y temporizadores.
- **Direcciones (Address)**: tipo utilizado para direccionar el SUT.

4.2.2 Componentes

El componente se define como un Tipo Componente dentro de los Tipos de Configuración vistos anteriormente. Dentro de él se define la estructura y propiedades del proceso o procesos (MTCs y PTCs) en los que se va a ejecutar el caso de prueba incluyendo los puertos de los que va a poder hacer uso cada proceso en tiempo de ejecución, así como las variables, constantes y temporizadores.

4.2.3 Plantillas (“Templates”)

Los plantillas se usan para definir los datos que van a ser usados en las pruebas. Las plantillas son contenedores de datos que tienen dos funciones sumamente importantes dentro de TTCN-3:

1. **Plantillas de envío**. Permiten preparar los datos de un mensaje antes de volcarlos al mismo para facilitar al probador esta labor.
2. **Plantillas de recepción**. Las plantillas de recepción permiten mecanismos de comparación, es decir, nos permiten comparar el mensaje que nos llega con una plantilla previamente preparada con los datos que el caso de prueba espera recibir. De este modo podemos comparar todo el mensaje o únicamente los datos que nos sean relevantes. Esta funcionalidad de comparación entre el mensaje que llega y la plantilla es parte del lenguaje TTCN-3 y muy sencilla para el programador, el cuál únicamente tiene que utilizar las macros SEND y RECEIVE de TTCN-3 pasando el mensaje de envío o recepción, y la operación se lleva a cabo de manera transparente.

Con la operación Value of se asigna los valores de la plantilla a una variable.

Al introducir datos a una plantilla están permitidos símbolos como:

- **?**: para plantillas de recepción, cualquier cosa que venga en ese dato será aceptada como chequeo positivo. Si ese campo llega vacío, dará error.
- *****: para plantillas de recepción, valida como positivo llegue o no llegue dato.
- **omit**: para plantillas de envío (no se envía ese dato) y recepción (se chequea que el campo llega vacío).
- **Complement**: lista de valores no admitidos en recepción.
- **Range**: rango de valores admitidos.

- **Permutation**: acepta valores de permutaciones de x elementos.
- **Length**: máxima longitud del elemento recibido.
- **Ifpresent**: compara el elemento si llega (si no llega omit).

Se presenta a continuación un ejemplo de una plantilla de recepción para el mensaje de ABORT indication. Con esta plantilla al recibir un ABORTind, se compararán automáticamente todos sus campos, esperando recibir algún dato (sin importar el valor) en los campos quality_of_Service y user_information y el valor exacto en dialogue_ID, abort_reason y application_context_name.

```
template ASP_TCAP_U_ABORTind t_U_ABORTind ( template integer p_dialogue_ID,
                                            template octetstring p_abort_reason,
                                            template objid p_application_context_name ) :=
{
  quality_of_Service := ?,
  dialogue_ID := p_dialogue_ID,
  abort_reason := p_abort_reason,
  application_context_name := p_application_context_name,
  user_information := ?
}
```

4.2.4 Puertos

Los puertos en TTCN-3 se pueden dividir en dos partes:

- Parte en TTCN-3 en la que se definen los tipos de mensaje de entrada y salida que van a admitir los puertos.
- Parte en C: esta parte es una API en la que se implementan el envío y recepción de cada mensaje que admite el puerto en cuestión.

4.2.5 Casos de Prueba

Cada caso de prueba se define con la palabra clave *testcase* y tiene un nombre identificativo único.

4.2.6 Funciones

No hay ninguna particularidad en las funciones en TTCN-3. Como en cualquier lenguaje, se definen funciones dentro de los ficheros .ttcn que posteriormente pueden ser llamadas desde otros puntos del mismo fichero u otros ficheros con visibilidad a esa función.

4.2.7 Comportamiento Alternativo (“Alt Step”)

En TTCN-3 se puede dejar el siguiente paso de la ejecución a expensas de un evento externo; para ello se utilizan los Alt Steps. Por ejemplo dependiendo de si me llega el mensaje A o el B ejecuto un trozo de código u otro.

4.2.8 Generación de un conjunto de pruebas ejecutables

Este gráfico muestra el flujo de generación de un conjunto de pruebas en TTCN-3:

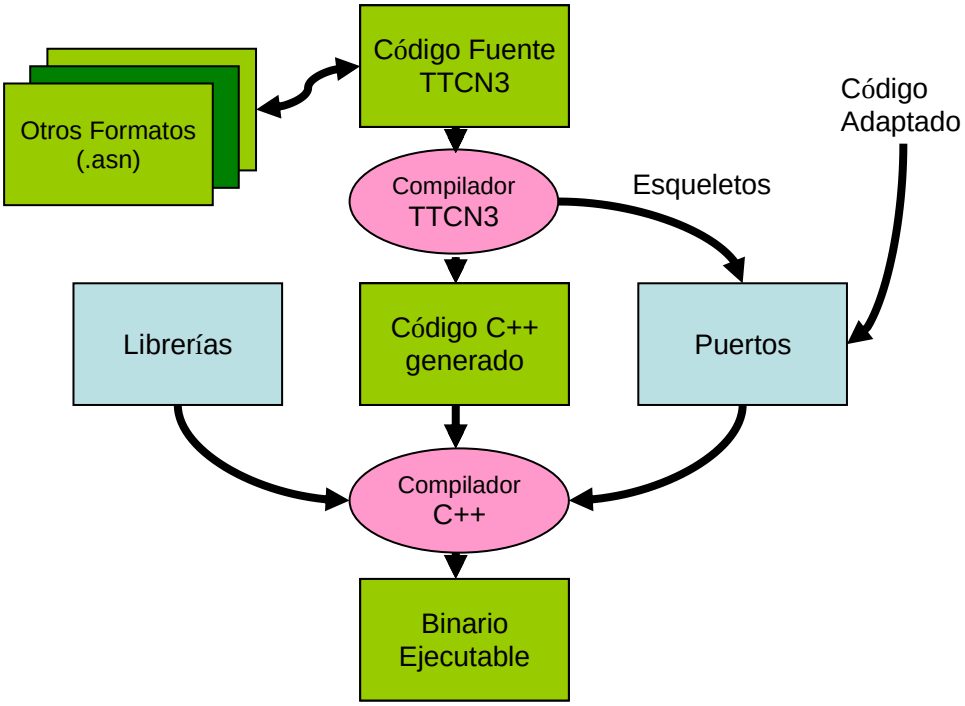


Figura 17 – Flujo de generación de código TTCN-3

Como se puede ver se realiza una primera compilación del código TTCN-3 que transforma el código en ficheros C++, con los que se realiza una segunda compilación de la que resulta el fichero ejecutable. En esta segunda compilación se puede añadir código escrito directamente en C++.

4.3 Operaciones definidas en TTCN-3

Existen Operaciones definidas en TTCN-3 muy útiles para manejar los MTC, PTC's y comunicar estos con el SUT.

Operaciones de Configuración:

- **create**: crea un PTC.
- **connect/disconnect**: conecta/desconecta un componente con otro.
- **map/unmap**: conecta/desconecta un puerto de un componente con otro del SUT.

- **mtc/system/self**: palabras clave para referirse a las direcciones del MTC, SUT y dirección propia del componente donde se use.
- **start/stop**: inicia/finaliza la ejecución de un componente.
- **running**: chequea si un PTC sigue en ejecución.
- **alive**: chequea si un PTC sigue vivo.
- **done**: espera la finalización de un PTC.
- **killed**: chequea la muerte de un PTC.

Operaciones de Comunicación:

Comunicación basada en mensajes:

- **send**: envía un mensaje.
- **receive**: recive un mensaje.

Comunicación basada en procedimientos:

- **call**: llamada a un procedimiento.
- **getcall**: aceptación de la llamada del procedimiento desde entidad remota
- **reply:** Respuesta de la llamada del procedimiento desde una entidad remota
- **raise:** lanzar una excepción
- **getreply:** gestionar la respuesta de una llamada previa
- **catch:** capturar una excepción

Operaciones relacionadas con los puertos:

- **check**: examinar msg/call/exception/reply recibida en el puerto.
- **clear**: limpiar el puerto de las operaciones encoladas.
- **start:** limpiar y dar acceso al puerto.
- **stop:** parar el puerto.

Operaciones de Temporizador:

- **start**: arrancar cuenta del temporizador.
- **stop**: parar la cuenta del temporizador.
- **read**: leer la cuenta del temporizador.
- **running**: chequear si el temporizador está corriendo.
- **timeout**: chequear si el temporizador ha llegado al máximo.

Operaciones de Veredicto:

Cada caso de prueba tiene un veredicto final, al cual se le puede ir dando valor en cualquier punto del código que se considere oportuno. Las operaciones para ello son:

- **setveredict**: fija el valor del veredicto
- **getveredict**: lee el valor del veredicto

El veredicto puede tomar los siguientes valores:

- **none**: estado inicial
- **pass**: caso pasado satisfactoriamente
- **fail**: caso no pasado satisfactoriamente
- **inconc**: caso sin concluir
- **error**: error durante la ejecución del caso

4.4 Código TTCN-3 común para las pruebas

Todas las pruebas implementadas en TTCN-3 para el HLR, tienen muchas cosas en común, por lo que se dispone de un código o "framework" común que es compilado y utilizado por todos los casos de prueba. Este código común comprende las siguientes definiciones relevantes:

- Definición del componente utilizado por el MTC y los PTC

```
type component TCAP_MML_Component
{
 port TCAPasp_PT  TCAP_HLR;
 port MMLasp_PT   MML_MTC_PORT;
 port MMLasp_PT   MML_MTC_PORT2;
 port MMLasp_PT   MML_MTC_PORT3;
 port MMLasp_PT   MML_MTC_PORT4;
 port MMLasp_PT   MML_MTC_PORT5;
 port MMLasp_PT   MML_MTC_PORT6;
 port MMLasp_PT   MML_MTC_PORT7;
 port MMLasp_PT   MML_MTC_PORT8;
 port MMLasp_PT   MML_MTC_PORT_TRACE_CMDS;
 port MMLasp_PT   MML_MTC_PORT_TRACE_DATA;
 }
```

- Implementación de los puertos
- Ficheros asn

```
UpdateGprsLocationArg ::= SEQUENCE {
  imsi          IMSI,
  sgsn-Number ISDN-AddressString,
  sgsn-Address    GSN-Address,
  extensionContainer  ExtensionContainer  OPTIONAL,
  ... ,
  sgsn-Capability [0] SGSN-Capability OPTIONAL,
  informPreviousNetworkEntity [1] NULL       OPTIONAL,
  ps-LCS-NotSupportedByUE [2] NULL       OPTIONAL,
  v-gmlc-Address  [3] GSN-Address OPTIONAL,
  add-info       [4]  ADD-Info   OPTIONAL }
```

- Defición de macros generales: eg: código de operación de los mensajes

```
const integer cg_MAP_operationCode_UpdateLocation  := 2;
const integer cg_MAP_operationCode_InsertSD  := 7;
const integer cg_MAP_operationCode_SendRoutingInfo := 22;
const integer cg_MAP_operationCode_ProvideRoamingNumber  := 4;
const integer cg_MAP_operationCode_CancelLocation  := 3;
```

- Templates reutilizables

```
template MML_PROCEDURE_PRINTOUT t_EXECUTED :=
```

```
{
  result    := EXECUTED,
  textbox1  := omit,
  faultCode := omit,
  textbox2  := omit
}
```

- Funciones útiles
  - Envío tráfico MAP

```
function f_send_BEGIN_INVOKE
function f_send_END_ERROR
```

  - Recepción tráfico MAP

```
function  f_receive_BEGIN_INVOKE
function f_receive_U_ABORT
```

  - Envío comandos MML

```
function f_MML_execute_cmd ( charstring  p_MML_Command,
  template MML_PROCEDURE_PRINTOUT p_MML_Procedure_Printout := omit,
  boolean p_setverdict := true,
  float p_timer := 15.0 )
runs on TCAP_MML_Component return boolean
```

4.5 Conclusiones

Hemos podido comprobar, TTCN-3 es un potente lenguaje orientado a pruebas con el que realizar la validación de sistemas software en multitud de entornos y nodos, especialmente en telecomunicaciones.

TTCN-3 permite reutilizar los casos de prueba ya programados infinitas veces, a coste casi cero, para validar las funcionalidades incluidas en ellos cada vez que se haga una integración del software (o sea requerido por otro motivo), algo imprescindible para poder utilizar una metodología de desarrollo Agile/Scrum.

Capítulo 5

Funcionalidad probada en el HLR

Dado que el objetivo del proyecto era sobre todo demostrar la potencia y versatilidad de las pruebas automatizadas mediante TTCN-3, hemos seleccionado para automatizar sus pruebas funcionalidades representativas del HLR y fácilmente comprensibles sin tener un conocimiento exhaustivo de la red de telefonía móvil.

La estructura que vamos a seguir es ir explicando una a una las funcionalidades, plantear los casos de prueba de cada funcionalidad, programar dichos casos de prueba en TTCN-3 y finalmente ejecutar dichas pruebas comprobando los logs resultantes.

Las dos primeras funcionalidades que probamos y vamos a explicar son las localizaciones del abonado en dos dominios distintos de la red de telefonía: Circuitos y Paquetes, pero antes vamos a ver como en la red coexisten estos dos dominios y la función de cada uno de ellos.

Dominio de Circuitos Conmutados: este dominio permite la funcionalidad básica de la telefonía realizar una llamada a través de un circuito, aunque también se incluyen otras nuevas como el mensaje de texto. De alguna forma se están migrando los servicios de la telefonía fija a la móvil con este dominio. Los nodos involucrados son el VLR/MSC y el GMSC.

Dominio de Paquetes: Nace de la necesidad de interactuar con Internet. La telefonía móvil siente la necesidad de comunicarse con el mundo IP y para ello necesita gestionar paquetes IP. Nace el dominio de paquetes con el GPRS. Para la gestión de paquetes se utilizan los nodos SGNS (realiza las funciones equivalentes al VLR/MSC en el dominio

de circuitos) y GGSN (realiza las funciones equivalentes al GMSC en el dominio de circuitos).

El siguiente gráfico ilustra los dos dominios dentro de la red de telefonía y sus interfaces, tanto con el terminal móvil como con las redes de circuitos y la red IP en la cual se sitúa Internet.

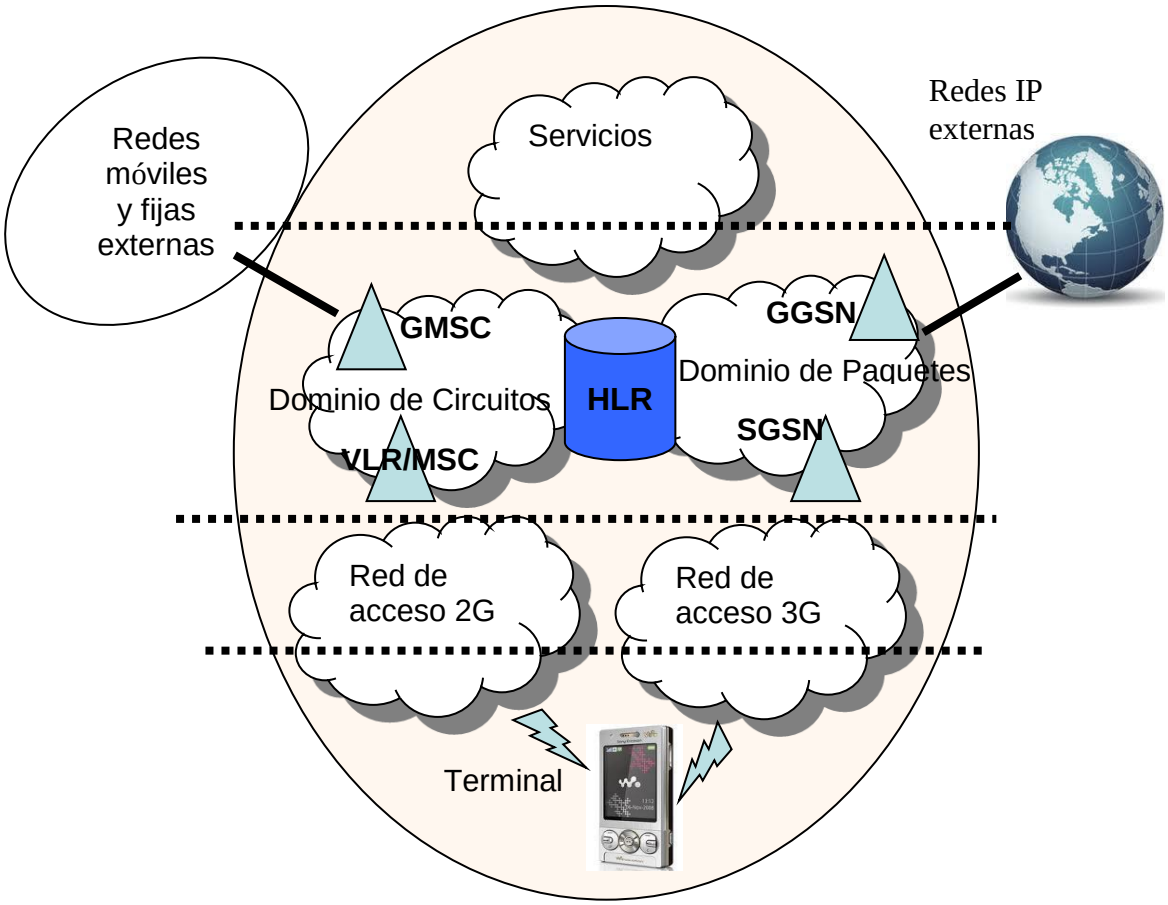


Figura 18 – División entre dominio de circuitos y dominio de paquetes

5.1 Update Location

Especificación técnica: 3GPP 29.002 - Mobile Applications Part (MAP)

El propósito de esta funcionalidad[ix] es la actualización de los datos de localización en el dominio de circuitos del abonado (MSRN o número de la MSC, LMSID si está disponible, y número de la VLR) en el HLR y la actualización del VLR con los datos de dicho abonado. Como su propio nombre indica (Registro Local de Visitantes), el VLR es un repositorio temporal de los datos de los abonado visitantes de ese área, cedidos por el HLR mientras está localizado en el área de dicha VLR. El LMSID viene del inglés "Local Mobile Station Identity" que se puede traducir como Identidad Local de la Estación Móvil. Este parámetro representa una identidad interna del móvil, reservada por el VLR para cada abonado registrado en él; se utiliza para ciertas operaciones en lugar del IMSI.

Los mensajes MAP utilizados en esta funcionalidad son:

- Mensaje ***Update Location*** (UL) (Actualizar Localización)
- Mensaje ***Insert Subscriber Data*** (ISD) (Insertar Datos de Abonado)
- Mensaje ***Cancel Location*** (CL) (Cancelar Localización)
- Forward ***Check SS Indication*** (FC-SS-I) (Comprobación de Indicación de Servicios Suplementarios)

Para la operación de Update Location debemos diferenciar entre las diferentes versiones de MAP, es decir, es distinta la funcionalidad que existe si esta operación se realiza utilizando el protocolo MAPv1 que si lo hace con MAPv2/v3. El diagrama de estados sí es común para del Update Location es:

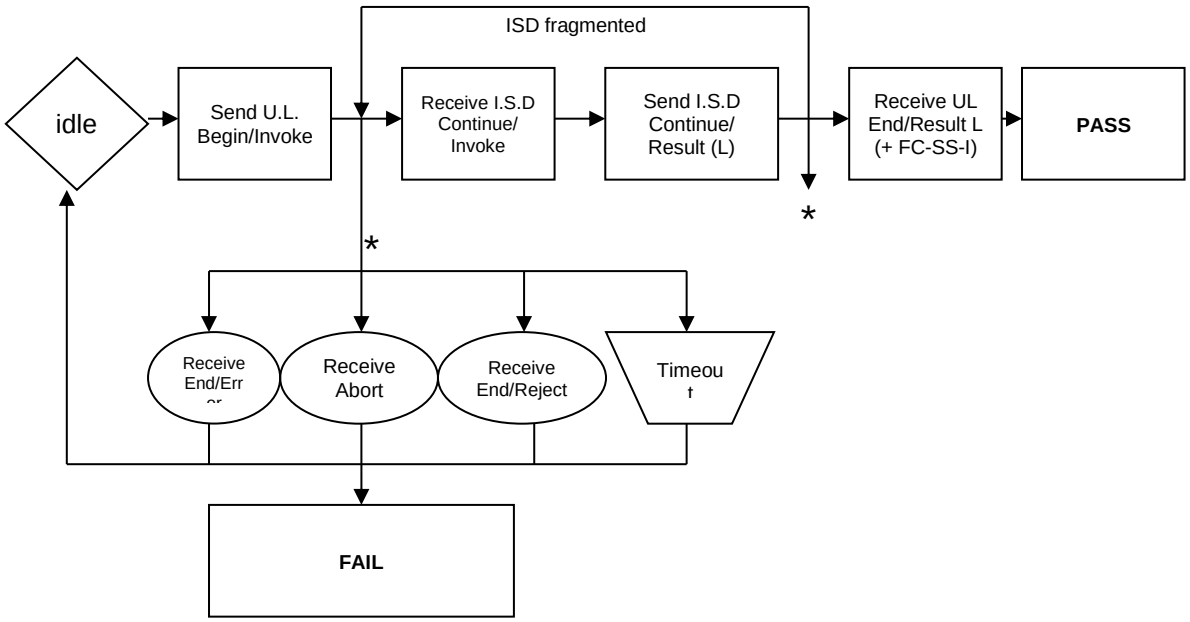


Figura 19 – Diagrama de estados del Update Location

5.1.1 Update Location MAP v1

Desde el punto de vista del HLR, el mensaje de Update Location siempre vamos a recibirlo desde un VLR. El HLR analiza el mensaje y dependiendo de ese análisis puede responder:

1. **Error:**

   1.1. *Abonado desconocido* (Unknown subscriber): el IMSI no está en el HLR.
   1.2. *Roaming no permitido* (Roaming not allowed): un abonado ha hecho roaming a un área restringida. Este error se puede producir por 3 causas:
      1.2.1. El número de VLR recibido corresponde a un área restringida para todos los abonados definida en el HLR. El abonado queda marcado como restringido.
      1.2.2. El abonado intenta localizarse en un área para el que está bloqueado su roaming. El abonado queda en estado bloqueado.
      1.2.3. El abonado tiene un servicio suplementario no soportado por el VLR en el que pretende localizarse y esto implica que no pueda producirse el roaming. El abonado queda marcado como restringido.
   1.3. *Valor inesperado de un parámetro* (unexpected data value): error en un parámetro.

2. **Caso Positivo**

   2.1. Si no se ha producido ningún error, el HLR reemplazará los datos de localización del abonado con los recibidos en el mensaje de Update Location, y si el número de VLR es distinto al anterior, cancelará la localización en el VLR en el que el abonado estaba localizado previamente, mandando un Cancel Location.
   2.2. El HLR manda uno/varios (dependiendo de la cantidad de datos) Insert Subscriber Data hacia el nuevo VLR para actualizar los datos del abonado que se acaba de localizar. Los datos incluidos en el ISD son:
      - Obligatorios:
         - MSISDN
         - Categoría
         - Estado del abonado
      - Opcionales:
         - Lista de Servicios Portadores es incluida en el mensaje si el abonado está subscrito a los servicios Portadores.
         - Lista de Teleservicios si el abonado posee alguno.
         - Servicios Suplementarios Provisionados si el abonado posee algún Servicio Suplementario relevante para el VLR.

Si el HLR recibe un error en respuesta al ISD desde el VLR, el procedimiento de localización acaba aquí.

Si hay respuesta positiva del VLR, el HLR comprueba si existe alguna marca de Forward Check SS para ese abonado, y en tal caso manda un Forward Check SS Indication además del mensaje de respuesta al Update Location. Si se ha mandado el

mensaje Forward Check SS Indication, se quita la marca de Forward Check SS de ese abonado.

5.1.2 Update Location MAP v2/v3

El mensaje de Update Location es recibido por el HLR desde un VLR. EL HLR analiza el mensaje y es capaz de responder con uno de los siguientes mensajes dependiendo de la información recibida y de la que él dispone:

1. **Error:**

   1.1. *Abonado desconocido* (Unknown subscriber): el IMSI no está en el HLR.
   1.2. *Roaming no permitido* (Roaming not allowed): un abonado ha hecho roaming a un área restringida. Este error se puede produce por 3 causas:
      1.2.1. El número de VLR recibido corresponde a un área restringida para todos los abonados definida en el HLR. El abonado queda marcado como restringido.
      1.2.2. El abonado intenta localizarse en un área para el que está bloqueado su roaming. El abonado queda en estado bloqueado (barred).
      1.2.3. El abonado tiene un servicio suplementario no soportado por el VLR en el que pretende localizarse y esto implica que no pueda producirse el roaming. El abonado queda marcado como restringido.
   1.3. *Valor inesperado de un parámetro* (Unexpected data value): error en un parámetro.
   1.4. *Datos Omitidos* (Data Missing): el mensaje de Update Location viene sin parámetros.

Si se produce uno de estos errores, el HLR cancelará la localización (envío mensaje Cancel Location) en el VLR en el que el abonado estaba localizado previamente a realizar el roaming.

Si no se ha producido ningún error de los descritos hasta aquí, el HLR sigue analizando el mensaje recibido y entramos a analizar el caso positivo de la localización en el dominio de circuitos:

2. **Caso Positivo**

   2.1. El HLR reemplazará los datos de localización del abonado con los recibidos en el mensaje de Update Location, y si el número de VLR es distinto al anterior, cancelará la localización en el VLR en el que el abonado estaba localizado previamente.
   2.2. El HLR manda uno/varios (dependiendo de la cantidad de datos) Insert Subscriber Data (ISD) hacia el nuevo VLR para actualizar los datos del abonado que se acaba de localizar. Los datos incluidos en el ISD son:
      - Obligatorios:
         - MSISDN
         - Categoría
         - Estado del abonado
      - Opcionales:

- Lista de Servicios Portadores que es incluida en el mensaje si el abonado está subscrito a los servicios Portadores.
- Lista de Teleservicios si el abonado posee alguno.
- Servicios Suplementarios Provisionados si el abonado posee algún Servicio Suplementario relevante para el VLR.
- ODB-Data: sólo se manda si el abonado tiene alguna prohibición. Los datos ODB permiten al operador establecer restricciones en las llamadas del abonado (Operador Determined Barring – Prohibiciones Determinadas por el Operador).
- Datos relacionados con otras funcionalidades: datos del abonado de alguna funcionalidad activa de la que el VLR necesite datos.

Si el VLR reporta que no soporta alguna funcionalidad que posee el abonado del que se ha mandado el ISD, y esto implica que el abonado no puede hacer roaming en este VLR, se genera un nuevo ISD con las restricciones del VLR y el abonado queda en el HLR estando en un área restringida.

Si se recibe un error como respuesta al ISD, el HLR mandará el error Fallo de Sistema (System Failure) como respuesta al Update Location.

Si hay respuesta positiva del VLR, el HLR comprueba si existe alguna marca de Forward Check SS para ese abonado, y en tal caso manda un Forward Check SS Indication además del mensaje de respuesta al Update Location. Si se ha mandado el mensaje Forward Check SS Indication, se quita la marca de Forward Check SS de ese abonado.

Las principales diferencias entre la operación Update Location en MAPv1 y MAPv2/v3 son los datos que se pueden incluir en el ISD desde el HLR, pudiendo contener ODB-Data y Datos extra en las versiones v2 y v3; la otra diferencia importante es que en v2 y v3 el VLR puede reportar que no soporta alguna de las funcionalidades que posee el abonado, generando el HLR un DSD para borrar del VLR los datos que están relacionados con dicha funcionalidad no soportada.

5.1.3 Manejo del mensaje Cancel Location

El HLR manda el mensaje Cancel Location para cancelar la localización de un abonado en un VLR. El HLR reintentará el envío si ocurre alguna de estas situaciones:

- Mensaje Cancel Location es devuelto por la red por una de estas causas:
  - Fallo de Subsistema
  - Congestión de Subsistema
  - Fallo de Red
  - Congestión de Red
- Limitación de recursos en el VLR
- Tiempo máximo de espera del mensaje de respuesta del VLR alcanzado.

El número de reintentos de envío del Cancel Location y el tiempo entre cada uno, se puede parametrizar a gusto del operador.

5.1.3.1 Forward Check SS Indication

Después de un reinicio del HLR tras un fallo, el VLR envía el campo Check SS dentro del Update Location a “chequeo requerido”. Tras el Update Location y envío de los datos de ese abonado al VLR, el HLR manda un Forward Check SS Indication al VLR y pone el campo Check SS a “chequeo no requerido”.

El Invoke de esta operación va junto con el Invoke del Resultado del Update Location.

5.1.3.2 Diagrama de Secuencia-Update Location

Diagrama de secuencia de un Error/Reject/Abort:

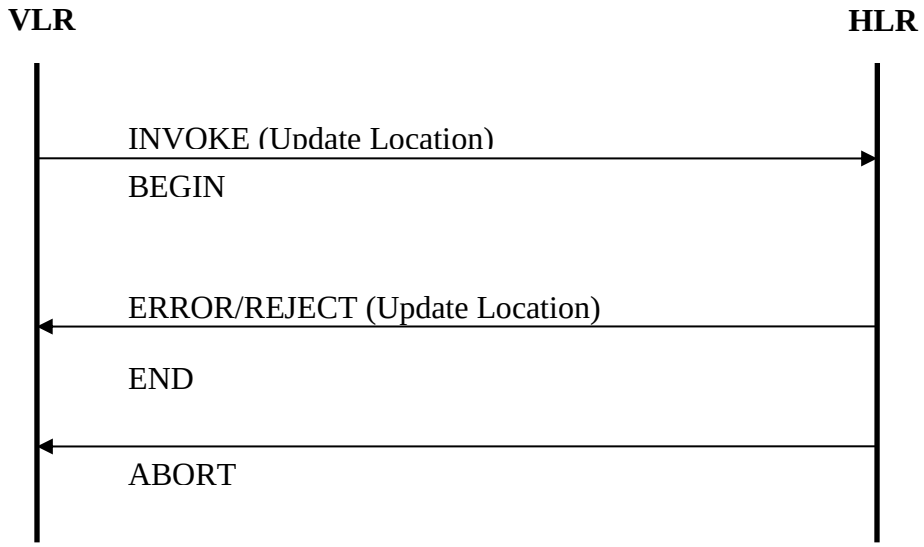


Figura 20 – Diagrama de secuencia de Error/Reject/Abort en Update Location

Diagrama de secuencia de un caso positivo:

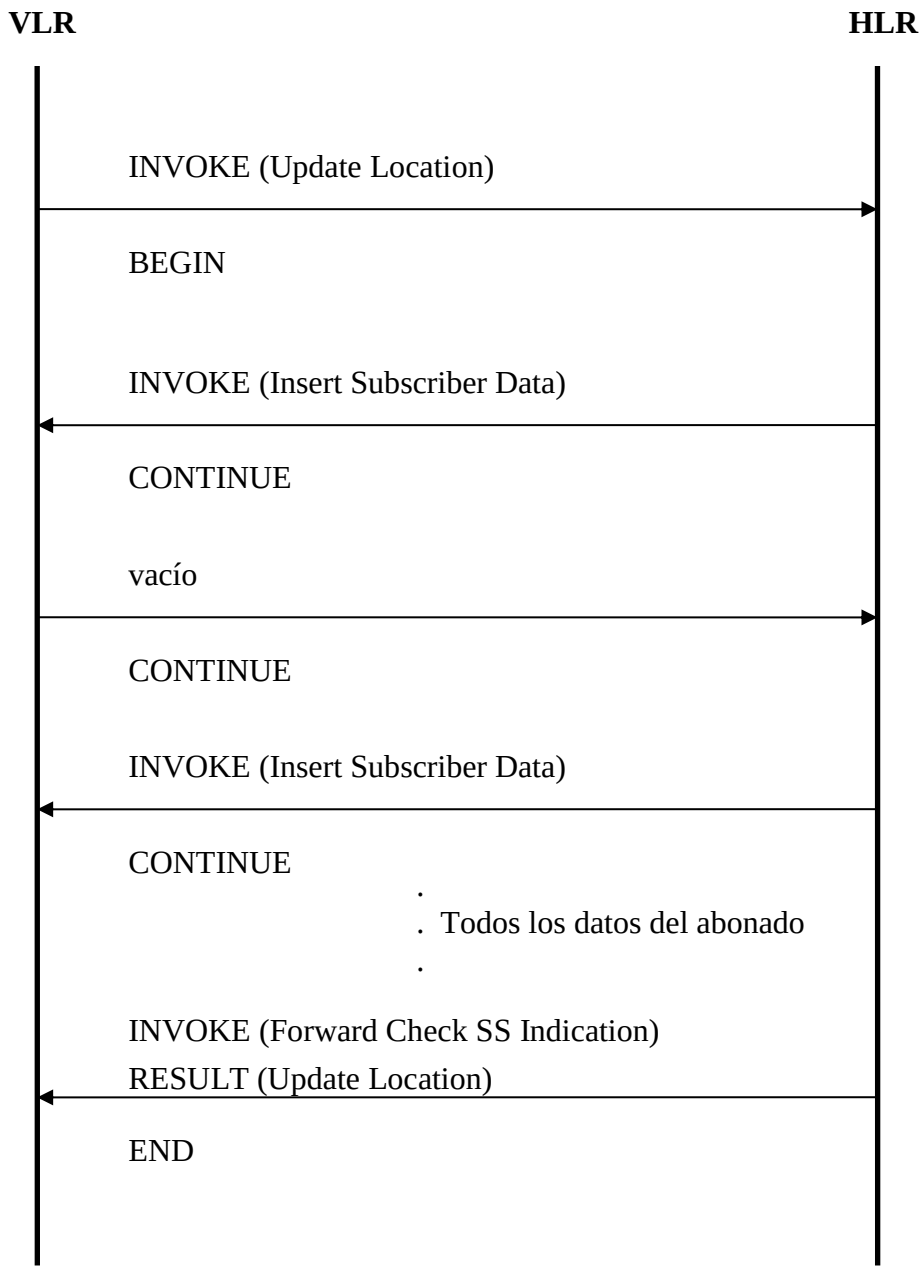


Figura 21 – Diagrama de secuencia de caso positivo en Update Location

5.2 GPRS Location Updating

Especificación técnica: 3GPP 29.002 - Mobile Applications Part (MAP)

Esta operación[x], sólo implementada en MAPv3, permite localizar al abonado en el dominio de paquetes. El HLR interactúa con el SGSN en este dominio, el cual es el encargado de enrutar los paquetes IP desde/hacia el Terminal móvil.

Desde el punto de vista del HLR la operación comienza con la llegada del mensaje GPRS Location Updating al HLR desde el nodo SGSN. Este mensaje es recibido por el HLR y es capaz de responder con uno de los siguientes mensajes dependiendo de la información recibida y de la que él dispone:

1. **Error:**

   1.1. *Abonado desconocido* (Unknown subscriber): se envía con diagnóstico "IMSI desconocido" cuando el IMSI no está en el HLR.
   1.2. *Abonado desconocido* (Unknown subscriber): se envía con diagnóstico "Subscripción GPRS desconocida" cuando el abonado tiene acceso únicamente a la red de circuitos y no de paquetes.
   1.3. *Roaming no permitido* (Roaming not allowed): un abonado ha hecho roaming a un área restringida. Este error se puede producir por 3 causas:
      1.3.1. El número de SGSN recibido corresponde a un área restringida para todos los abonados definida en el HLR. El abonado queda marcado como restringido.
      1.3.2. El abonado intenta localizarse en un área para el que está bloqueado su roaming. El abonado queda en estado bloqueado (barred).
      1.3.3. El abonado tiene una funcionalidad no soportada por el SGSN en el que pretende localizarse y esto implica que no pueda producirse el roaming. El abonado queda marcado como restringido.
   1.4. *Valor inesperado de un parámetro* (Unexpected data value): error en un parámetro.
   1.5. *Datos Omitidos* (Data Missing): el mensaje de Update Location viene sin parámetros.

Si se produce uno de estos errores, el HLR cancelará la localización (envío mensaje Cancel Location) en el SGSN en el que el abonado estaba localizado antes de producirse el roaming.

Si no se ha producido ningún error de los descritos hasta aquí, el HLR sigue analizando el mensaje recibido y entramos a analizar el caso positivo de la localización en el dominio de paquetes:

2. <u>Caso Positivo</u>

2.1. El HLR reemplazará los datos de localización del abonado con los recibidos en el mensaje de GPRS Location Updating, y si el número de SGSN es distinto al anterior, cancelará la localización en el SGSN en el que el abonado estaba localizado previamente.

2.2. El HLR manda uno/varios (dependiendo de la cantidad de datos) Insert Subscriber Data hacia el nuevo SGSN para actualizar los datos del abonado que se acaba de localizar. Los datos incluidos en el ISD son:

- Obligatorios:
  - MSISDN
  - NAM (Network Access Mode – Modo de Acceso a la Red): Con esta propiedad del abonado se le puede dotar de acceso a la red de Circuitos, de Paquetes o a ambas, los valores y su significado son:
    - NAM = 0 → Acceso a la red de Circuitos y Paquetes (red GPRS y red no-GPRS)
    - NAM = 1 → Acceso únicamente a la red de Circuitos (red no-GPRS)
    - NAM = 2 → Acceso únicamente a la red de Paquetes (red GPRS)
  - Estado del abonado
- Opcionales:
  - Lista de Servicios Portadores es incluida en el mensaje si el abonado está subscrito a los servicios Portadores.
  - Lista de Teleservicios si el abonado posee alguno.
  - ODB-Data: sólo se manda si el abonado tiene alguna prohibición. Los datos ODB permiten al operador establecer restricciones en las llamadas del abonado (Operador Determined Barring – Prohibiciones Determinadas por el Operador).
  - Lista de contextos PDP (Policy Decision Point): se envían los datos de los contextos PDP que están asignados a ese abonado. Un contexto PDP son las reglas que se van a seguir en el envío recepción de paquetes entre la red y el abonado (básicamente una calidad de servicio).
  - Datos relacionados con otras funcionalidades: datos del abonado de alguna funcionalidad activa de la que el SGSN necesite datos.

Si el SGSN informa que no soporta alguna funcionalidad que posee el abonado del que se ha mandado el ISD, y esto implica que el abonado no puede hacer roaming en este SGSN, se genera un nuevo ISD con las restricciones del SGSN y el abonado queda en el HLR estando en un área restringida.

Si se recibe un error como respuesta al ISD, el HLR mandará el error "Fallo de Sistema" (System Failure) como respuesta al GPRS Update Location.

Para la terminación exitosa de la operación, el HLR recibe la respuesta positiva al Insert Subscriber Data procedente del SGSN y manda el Result al Update GPRS Location.

5.2.1 Diagrama de Secuencia-GPRS Updating Location

Diagrama de secuencia de un Error/Reject/Abort:

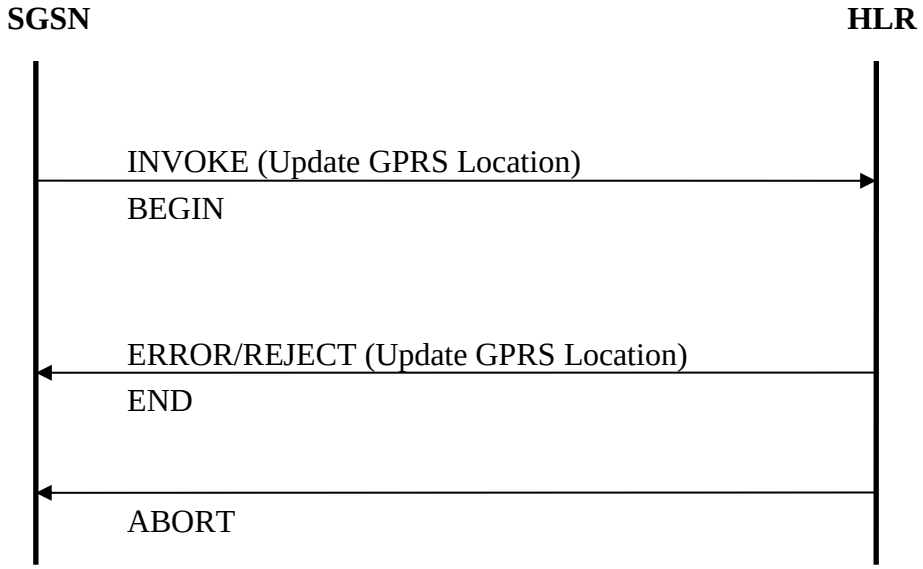


Figura 22 – Diagrama de secuencia de Error/Reject/Abort en Update GPRS Location

Diagrama de secuencia de un caso positivo:

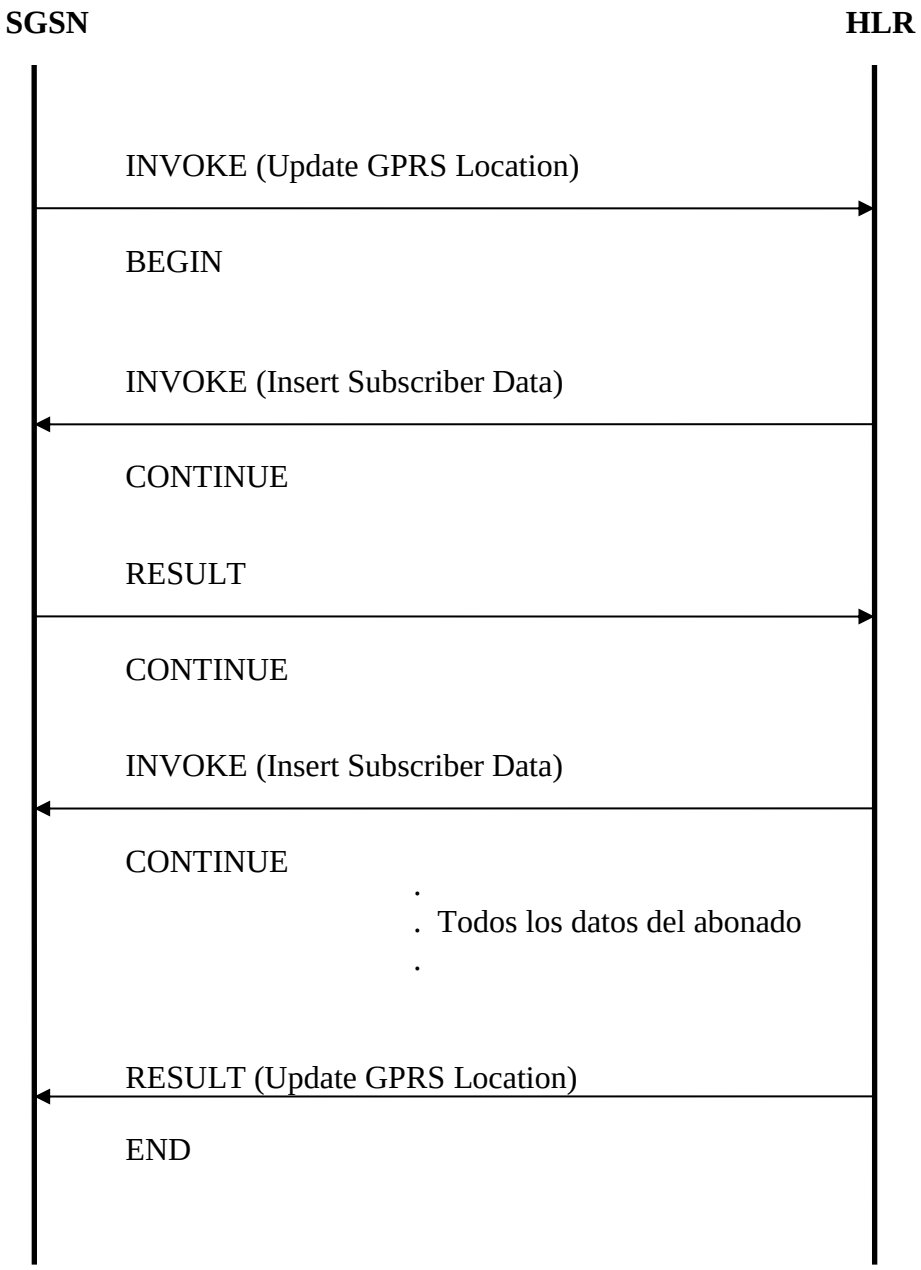


Figura 23 – Diagrama de secuencia de caso positivo en Update Location

5.3 Cambio de IMSI (IMSI Changeover)

La funcionalidad[xi] de IMSI Changeover permite a los operadores GSM/WCDMA cambiar las tarjetas SIM/USIM en sus terminales.

El proveedor de servicios puede realizar en cualquier momento un procedimiento de cambio de IMSI. El cambio se inicia mediante un comando que incluye: antiguo IMSI, nuevo IMSI y fecha de expiración (opcional).

Si se especifica fecha del cambio, el antiguo IMSI seguirá siendo usado hasta esa fecha en que se desencadenará automáticamente el cambio. Si no se especifica fecha, el cambio se realizará en el momento en que se introduzca el comando.

Desde el momento en que se realiza en cambio de IMSI, si llega alguna operación para el antiguo IMSI, esta será rechazada.

La fecha de expiración puede ser cambiada o ser ejecutado el cambio inmediatamente. También puede ser cancelado cualquier cambio de IMSI programado antes de que llegue la fecha de expiración.

Existe un atributo del abonado llamado estado del cambio de IMSI, el cual puede tener los siguientes valores:

- **PENDIENTE**: desde el momento que se programa un cambio de IMSI hasta que llega la fecha de expiración y éste tiene lugar.

- **FORZADO**: el cambio se ha ejecutado forzándolo por comando.

- **EJECUTADO**: el cambio ha tenido lugar en la fecha de expiración.

El diagrama de estados del cambio de IMSI puede ser el siguiente:

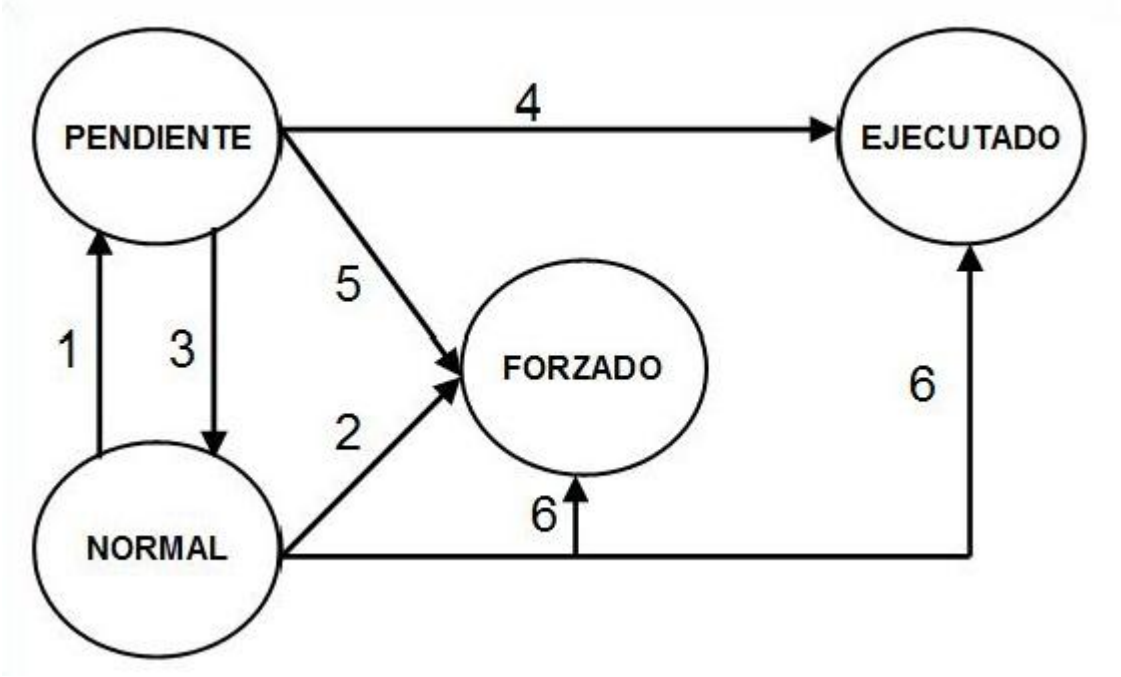


Figura 24 – Diagrama de estados del cambio de IMSI

1. Se programa un cambio de IMSI con fecha de expiración.
2. Se realiza un cambio de IMSI de ejecución inmediata (nuevo IMSI activo).
3. Se anula un cambio de IMSI programado para una fecha futura.
4. Se realiza el cambio de IMSI debido a que el abonado usa la nueva IMSI/USIM por primera vez antes de llegar a la fecha programada.
5. Se puede producir por 2 situaciones:
    a. Se fuerza a realizar inmediatamente un cambio de IMSI que estaba programado para una fecha futura.
    b. Llega la fecha de expiración y se produce el cambio de IMSI.
6. Por comando se ordena eliminar todas las referencias al antiguo IMSI.

Mientras el cambio de IMSI está en el estado **PENDIENTE**, éste será ejecutado si se recibe una de las siguientes operaciones para el nuevo IMSI:

a) Update Location
b) Note MS Present
c) Ready for Short Message
d) Begin Subscriber Activity
e) Un procedimiento de servicio suplementario en MAPv2 desde el VLR
f) Update GPRS Location

En estos casos el estado pasa a ser **EJECUTADO**.

Si se alcanza la fecha de expiración, el cambio se ejecuta en la siguiente operación de provisionado o tráfico que llegue para ese abonado. El cambio se realiza antes de realizar dicha operación, por lo que si ésta es para el antiguo IMSI será rechazada.

Las siguientes operaciones son aceptadas tanto para el antiguo como para el nuevo IMSI y no desencadenarán el cambio de IMSI:

a) Send Parameters MAPv1
b) Send Authentication Info MAPv2
c) Send Authentication Info MAPv3

Cuando se ejecuta un cambio de IMSI, la localización del abonado se pone a "desconocida" y se desencadena un Cancel Location hacia el VLR y/o SGSN donde se hallase localizado.

5.4 Servicio suplementario de rechazo de llamadas anónimas

El servicio suplementario de rechazo de llamadas anónimas[xii] (ACR- Anonymous Call Rejection) permite a los abonados móviles el rechazo de llamadas entrantes que no muestren el calling party number. Gracias a esta funcionalidad los abonados pueden prevenir ser molestados por llamadas indeseadas.

En el HLR existe un código USSD asociado a esta funcionalidad, cuyo propósito es el manejo de la activación/desactivación del servicio suplementario de rechazo de llamadas anónimas por procedimiento de abonado.

El USSD (Unstructured Supplementary Service Data - Servicio Suplementario de Datos no Estructurados) es un servicio para el envío de datos desde/hacia el terminal móvil. Estos mensajes no pueden ser almacenados, si no llegan a su destino se descartan y deben ser enviados de nuevo. Hay dos versiones o fases de mensajes USSD:

- La versión de USSD conocida como Fase 1, definida en el estándar GSM 02.90, sólo soporta operaciones de comunicación iniciadas por el teléfono, llamadas operaciones pull.
- La versión de USSD conocida como Fase 2, definida en el estándar GSM 03.90, soporta operaciones de comunicación iniciadas tanto por el teléfono como por la red, llamadas operaciones pull y push.

Si un abonado móvil tiene activada esta funcionalidad y le llega una llamada entrante anónima, el GMSC o gsmSCF es informado de que la llamada debe ser rechazada durante la recepción de la operación Send Routing Information MAPv3.

Acciones posibles sobre el servicio suplementario de rechazo de llamadas anónimas:

- **Provisionamiento** y **desabastecimiento**: lo realiza el Operador de Servicios mediante comando MML.

- **Activación** y **desactivación**: puede ser realizado por el Operador de Servicios mediante comando MML o por el abonado mediante procedimiento de abonado.

- **Apelación**: La apelación del servicio se realiza por el GMSC, el gsmSCF o el HLR. En el caso del HLR, se maneja la apelación cuando el abonado está provisto

del servicio de rechazo de llamadas anónimas junto con el servicio CLIP con sobreescritura; en este caso la invocación se realiza no mandando el servicio de rechazo de llamadas anónimas al GMSC/gsmSCF, permitiendo que la llamada se lleve a cabo.

5.4.1 Transferencia de los datos del Servicio Suplementario al GMSC o gsmSCF

El servicio suplementario de rechazo de llamadas anónimas es enviado por el HLR en el Send Routing Information MAPv3 cuando se cumplen las siguientes condiciones:

- Un Send Routing Info MAPv3 es recibido y aceptado por el HLR.
- El abonado al que va dirigida esa llamada tiene provisto y activado el servicio suplementario de rechazo de llamadas anónimas.

5.4.2 Procedimientos de Control de ACR por el abonado

Los abonados pueden activar y desactivar el suplementario de rechazo de llamadas anónimas por medio de códigos USSD. Veamos como sucede: cuando un abonado inicia el suplementario de rechazo de llamadas anónimas con un código USSD, éste llega al gestor de códigos USSD del HLR en el mensaje PROCESS UNSTRUCTURED SS DATA MAPv1/v2 y analiza el código para saber de qué aplicación USSD se trata. Si este proceso falla se reporta un error para indicar el código malformado o en caso de que se reconozca el ACR, se inicia la comprobación de si el servicio está provisto para ese abonado; en caso negativo el HLR devuelve un error, pero en caso de que si estuviera provisto se toma la acción correspondiente (activación/desactivación) y se reporta el éxito de la operación.

Los formatos de los procedimientos de abonado son:

1. *XY(Z)c#
2. #XY(Z)c#

Donde:

- X, Y, Z es un número entre 0 y 9. (Z es opcional) e identifican el código del servicio que se invoca.
- c es un parámetro opcional que se ignora al ser recibido.

Capítulo 6

Pruebas Realizadas

En este capítulo se describen las pruebas de funcionalidad realizadas al HLR mediante TTCN-3 durante este proyecto. Las pruebas realizadas sirven para verificar que cada funcionalidad está correctamente implementada en el HLR. En cada caso de prueba hay una primera parte de preparación del nodo para poder ejecutar la prueba, incluyendo la creación del abonado que se va a utilizar en caso de que la prueba requiera alguno. La segunda parte es la de la ejecución de la prueba en sí misma. Y en la tercera parte se realizan las acciones necesarias para dejar el nodo en el estado inicial. En las pruebas descritas a continuación los comandos generados por operadores y las acciones de los distintos nodos, excepto el HLR que es el nodo que se está probando, son definidos y generados mediante TTCN-3. Es decir, hay una plantilla TTCN-3 que define cada una de las acciones que vamos a describir en las pruebas, y también las verificaciones que se hacen a las respuestas del HLR para ver que funciona correctamente.

Es conveniente decir que todos los nodos para pruebas funcionales son preparados con unos datos de configuración estándar del HLR para pruebas funcionales. Estos datos incluyen desde la activación de las funcionalidades del HLR hasta la definición de los nodos externos, pasando por datos como la definición de las áreas, direcciones de enrutamiento, etc.…

6.1 Pruebas para la funcionalidad de Update Location

Caso de prueba 1 - Update Location - Abonado Desconocido

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.

Abonado/s:
1 abonado no definido en el HLR.

Preparación de la prueba desde el estado inicial

No es necesaria ninguna preparación previa a la ejecución de la prueba.

Prueba

Utilizando el VLR creado en TTCN-3 vamos a formar y enviar un mensaje BEGIN/INVOKE con la operación Update Location en MAPv1.

Este mensaje de actualización de la localización va a llegar al HLR, el cual lo analiza y comprueba que no tiene definido en su base de datos el abonado para el cuál se le pide la operación, por lo que responde con un mensaje END/ERROR en el que la causa del error será "*Abonado Desconocido*". El PTC que simula el VLR recibe este mensaje y comprueba que es un mensaje END/ERROR y que la causa es la correcta (la comprobación es parte dela plantilla TTCN-3).

Recuperación del estado inicial

No es necesaria ninguna acción para recuperar el estado inicial.

Código TTCN-3 del caso de prueba 1

Se crea un MTC con la palabra clave *testcase* llamado *Caso1_Update_location*, el cuál va a ser del tipo *TCAP_MML_Component*, definido en el código común como una entidad con varios puertos MML y TCAP:

```
type component TCAP_MML_Component {
  port TCAPasp_PT  TCAP_HLR;
  port MMLasp_PT   MML_MTC_PORT;
  port MMLasp_PT   MML_MTC_PORT2;
  port MMLasp_PT   MML_MTC_PORT3;
  port MMLasp_PT   MML_MTC_PORT4;
  port MMLasp_PT   MML_MTC_PORT5;
  port MMLasp_PT   MML_MTC_PORT6;
  port MMLasp_PT   MML_MTC_PORT7;
  port MMLasp_PT   MML_MTC_PORT8;
  port MMLasp_PT   MML_MTC_PORT_TRACE_CMDS;
  port MMLasp_PT   MML_MTC_PORT_TRACE_DATA;
  port MMLasp_PT   MML_MTC_PORT_DB[20];
  var integer      numPort := 1;

}
```

Dentro del MTC, únicamente se crea un PTC llamado TCAP_MML_Component_PTC y con los TEST PORTS PARAMETERS definidos para VLR_1_1106 y en ese PTC se inicia la función f_Caso_1_Update_Location que es en la que se va a introducir todo el código de la prueba. Esto se hace así para no utilizar los puertos del MTC y poder ejecutar casos de distintas funcionalidades en la misma ejecución.

Con la instrucción .done se espera a que el PTC acabe la ejecución de la función antes de acabar con el MTC.

```
testcase Caso1_Update_Location () runs on TCAP_MML_Component
   {

        var TCAP_MML_Component TCAP_MML_Component_PTC :=
TCAP_MML_Component.create("VLR_1_1106");
        TCAP_MML_Component_PTC.start ( f_Caso_1_Update_Location () );
        TCAP_MML_Component_PTC.done;
```

En la función f_Caso_1_Update_Location se va a lanzar la prueba y más PTCs en caso de ser necesarios. Comentar que el tipo del PTC es el mismo que el MTC: TCAP_MML_Component, por lo que posee los mismos puertos que el anterior, pero en este caso sí que se van a utilizar.

```
   function f_Caso_1_Update_Location () runs on TCAP_MML_Component
   {

        //Declaración de Variables.
        var boolean v_result := false;
        var MAP_Invoke v_mapInvoke_UL_v1;
        var TCAP_PAR_Address v_destination_address;
        var TCAP_PAR_Address v_originating_address;
        var octetstring v_Msisdn;
        var octetstring v_Imsi;
        var octetstring v_VLR_Number;

        //Mapeamos los puertos TCAP y MML contra el SUT.
        map ( self:TCAP_HLR, system:TCAP_HLR );
        map ( self:MML_MTC_PORT, system:MML_MTC_PORT );

        //Se ejecutan algunos commandos de preparación contra el HLR
        f_Manual_Preparations();

        //Convertimos varios variables al tipo adecuado para su uso.
      v_Msisdn := f_Converter_Charstring_To_Map_wPrefix ( px_user_1106.msisdn );
        v_Imsi := f_Converter_Charstring_To_Map ( "444444111111111" );
```

```
        v_VLR_Number := f_Converter_Hexstring_To_Map_wPrefix (
px_nodeOrigination_1106.number );

        v_destination_address := valueof ( t_TcapAddress ( ( int2bit (
px_nodeDestination_1106.PC,14 ) ), cg_HLR_SSN, px_nodeDestination_1106.number )
);
        v_originating_address := valueof ( t_TcapAddress ( ( int2bit (
px_nodeOrigination_1106.PC,14 ) ), cg_VLR_SSN, px_nodeOrigination_1106.number )
);
```

//Se crea un INVOKE con el template t_MAP_UpdateLocationInfoArg al que se le ha pasado los valores del invoke ID, el imsi y el vlr. Este template está definido en el código común como:

```
template MAP_Invoke t_MAP_UpdateLocationInfoArg ( integer p_invokeId, IMSI p_imsi,
ISDN_AddressString p_msc_Number ) :=
{
  invokeId := { present_ := p_invokeId },
  linkedId:= { absent :=NULL },
  opcode := { local := cg_MAP_operationCode_UpdateLocation },
  argument:=
  {
    updateLocationArg:=
    {
      imsi := p_imsi,
      msc_Number := p_msc_Number,
      vlr_Number := p_msc_Number,
      lmsi := omit,
      extensionContainer := omit,
      vlr_Capability      := omit,
      informPreviousNetworkEntity := omit,
      cs_LCS_NotSupportedByUE     := omit,
      v_gmlc_Address      := omit,
      add_info := omit
    }
  }
}
```

```
        v_mapInvoke_UL_v1 := valueof ( t_MAP_UpdateLocationInfoArg (
px_Invoke_ID_1106, v_Imsi, v_VLR_Number ) );
```

//Se utiliza la función común f_send_BEGIN_INVOKE para mandar el BEGIN INVOKE contra el HLR. Se le debe pasar como parámetros el dialogue ID, invoke ID, dirección de destino y de origen, código de la operación MAP del Update Location, el rango de clase, el invoke que he construido antes y true o false dependiendo si se quiere que se codifique el invoke o no. Remarcar que esta función ya hace uso directamente de los puertos TCAP del PTC con la instrucción .send. La función está codificada de la siguiente manera en el código común:

```
  function f_send_BEGIN_INVOKE ( template integer p_Dialogue_ID, template integer
p_Invoke_ID, template objid p_Application_Context_Name,
    template TCAP_Address p_Destination_Address, template TCAP_Address
p_Originating_Address, template integer p_Operation,
    template TCAP_ClassRange p_Class, template MAP_Invoke p_Map_Invoke, boolean p_Encode )
  runs on TCAP_MML_Component
  {
    if ( p_Encode == false ) {
      TCAP_HLR.send ( t_INVOKEreq ( p_Dialogue_ID, p_Class, p_Invoke_ID, p_Operation, omit
) );
      log("MAP: INVOKE sent:\n",t_INVOKEreq ( p_Dialogue_ID, p_Class, p_Invoke_ID,
p_Operation, omit ));
    }
    else {
      TCAP_HLR.send ( t_INVOKEreq ( p_Dialogue_ID, p_Class, p_Invoke_ID, p_Operation,
enc_MAP_Invoke ( valueof ( p_Map_Invoke ) ) ) );
      log("MAP: INVOKE sent:\n",t_INVOKEreq ( p_Dialogue_ID, p_Class, p_Invoke_ID,
p_Operation, enc_MAP_Invoke ( valueof ( p_Map_Invoke ) ) ));
    }

    TCAP_HLR.send ( t_BEGINreq ( p_Destination_Address, p_Application_Context_Name,
p_Originating_Address, p_Dialogue_ID ) );
    log("MAP: BEGIN sent:\n",t_BEGINreq ( p_Destination_Address,
p_Application_Context_Name, p_Originating_Address, p_Dialogue_ID ));
    setverdict ( pass );
```

```
  }

      f_send_BEGIN_INVOKE ( px_Dialogue_ID_1106, px_Invoke_ID_1106, omit,
v_destination_address, v_originating_address,
cg_MAP_operationCode_UpdateLocation, succAndFailReported, v_mapInvoke_UL_v1,
true );

//Ahora se debe recibir el mensaje END/ERROR procedente del HLR. Para ello se
hace uso de otra función común, f_receive_END_ERROR, a la que se le pasa como
parámetros el dialogue ID, invoke ID, application contextname, el código de
error, un template del Return Error, true o false para pedir que decodifique el
invoke y un timer con el tiempo de espera máximo. Esta función está implementada
de la siguiente manera en el código común:

  function f_receive_END_ERROR ( template integer p_Dialogue_ID, template integer
p_Invoke_ID, template objid p_Application_Context_Name,
    template integer p_Error_Value, template MAP_ReturnError p_Map_Return_Error, boolean
p_Decode, float p_Timer_Value )
  runs on TCAP_MML_Component return Boolean
  {
    var boolean v_result := true;
    var ASP_TCAP_ENDind v_ENDind;
    var ASP_TCAP_U_ERRORind v_ERRORind;
    var MAP_ReturnError v_MapReturnError;

    timer t_timer := p_Timer_Value;
    t_timer.start;

    // RECEIVE ENDind + ERRORind.
    alt
    {
      [] TCAP_HLR.receive ( t_ENDind ( p_Dialogue_ID, p_Application_Context_Name ) ) ->
value v_ENDind
      {
        log ( "MAP: END received:\n", v_ENDind );
        setverdict ( pass );
        v_result := true;
        alt
        {
          [] TCAP_HLR.receive ( t_U_ERRORind ( p_Dialogue_ID, p_Invoke_ID, p_Error_Value )
) -> value v_ERRORind
          {
            log ( "MAP: ERROR received:\n", v_ERRORind );
            setverdict ( pass );
            v_result := true;
          }
          [] as_AnyTcapOrTimeout ( t_timer )
          {
            setverdict ( fail , "ERROR: MAP: ERROR message expected." );
            v_result := false;
            return v_result;
          }
        }
      }
      [] as_AnyTcapOrTimeout ( t_timer )
      {
        setverdict ( fail , "ERROR: MAP: END message expected." );
        v_result := false;
        return v_result;
      }
    }

    if ( p_Decode == true ) {
      // DECODE MAP_ERROR
      if( isvalue(p_Application_Context_Name) ) {
        v_result := f_decodeMAPReturnError ( v_ERRORind, v_MapReturnError,
valueof(p_Application_Context_Name) );
      }
      else {
        v_result := f_decodeMAPReturnError ( v_ERRORind, v_MapReturnError, omit );
      }

      if ( v_result )
      {
        //MATCH
```

```
        v_result := match ( v_MapReturnError, p_Map_Return_Error );

        if ( not v_result )
        {
          log ("ERROR: Mismatch: MAP_ERROR, match details:");
          log ( match ( v_MapReturnError, p_Map_Return_Error ) );
          setverdict ( fail );
          v_result := false;
          return v_result;
        }
        else
        {
          log ( "MAP: Message matched" );
        }
      }
    }

    return v_result;
  }
```

```
      v_result := f_receive_END_ERROR ( px_Dialogue_ID_1106, px_Invoke_ID_1106,
?, cg_MAP_errorCode_UnknownSubscriberParam,
t_MAP_UnknownSubscriberParam(px_Invoke_ID_1106), false, px_Timer_Value_1106 );

//Si no ha ido bien la recepción, se loguea el error en la función y se salta a
la etiqueta END para finalizar el caso de prueba.

        if (not v_result)
        {
            log("Error: f_receive_END_ERROR");
        }

//Se restauran las preparaciones realizadas en el HLR para dejar el nodo en el
mismo estado que tenía antes del caso de prueba.

        f_Restore_Manual_Preparations();

      //Desmapeamos los puertos TCAP y MML
      unmap ( self:TCAP_HLR, system:TCAP_HLR );
      unmap ( self:MML_MTC_PORT, system:MML_MTC_PORT );
    }
```

Caso de prueba 2 - Update Location – Update Location sobre un abonado con AOC desde un VLR que no lo soporta - Abonado Restringido

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3. Situado en un área que soporta AOC.
- **HLR** (SSN = 6) – Nodo bajo prueba utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en los mismos PTCs de TTCN-3 que los VLRs.

Abonado/s:
1 abonado definido durante la prueba en el HLR.

Preparación de la prueba desde el estado inicial

En este caso de prueba definimos un abonado (mediante comandos del operador simulado con TTCN-3) en el HLR y se le activa el servicio básico de telefonía con el que puede realizar y recibir llamadas. Después le proveemos con el servicio de "aviso de facturación".

***Servicio de aviso de facturación (AOC)*:** este servicio da al abonado la información necesaria para calcular el coste de una llamada que va a realizar. Este servicio es únicamente aplicable si se utiliza MAPv1 como protocolo.

En esta prueba configuramos en el HLR el área de nuestro VLR (identificada por la dirección del mismo) como área que no soporta AOC. Esta área o dirección de VLR a la que hacemos referencia ha sido definida previamente en los datos de configuración estándar del HLR para pruebas funcionales.

Prueba

Desde el VLR de nuestro caso (definido en el HLR como VLR que no soporta AOC) enviamos un mensaje de Update Location en MAPv1. Cuando el HLR recibe un Update Location para un abonado que posee AOC desde un VLR que no lo soporta, el abonado se marca como Restringido y se recibe como respuesta un mensaje de error con el error "Roaming No Permitido". Comprobaremos que la prueba ha sido satisfactoria analizando el mensaje de respuesta entre el HLR y el VLR y chequeando que recibimos dicho error con una plantilla previamente definida en TTCN.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba, restauramos las características del área en el HLR y borramos el abonado (mediante comandos del operador), recibiendo el correspondiente mensaje MAP de Cancel Location desde el HLR al VLR para informar

al VLR que dicho abonado ya no debe figurar como localizado en sus datos (comportamiento esperado que es verificado en la plantilla TTCN-3).

Caso de prueba3 - Update Location – Update Location desde un área restringida a una que no lo está

Elementos Involucrados

Nodo/s:

- **VLR1** (SSN = 7) – Simulado en un PTC de TTCN-3. Situado en un área que no soporta Roaming.
- **VLR2** (SSN = 7) – Simulado en un PTC de TTCN-3. Situado en un área que sí soporta Roaming.
- **HLR** (SSN = 6) – Nodo levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en los mismos PTCs de TTCN-3 que los VLRs.

Abonado/s:
1 abonado definido durante la prueba en el HLR.

Preparación de la prueba desde el estado inicial

En esta prueba definimos un abonado en el HLR y le activamos el servicio básico de telefonía con el que puede realizar y recibir llamadas.

A continuación configuramos en el HLR dos áreas distintas de VLRs (identificadas por la dirección de cada uno de los mismos), una como área que soporta Roaming y la otra como área que no lo soporta. Estas áreas o direcciones de VLRs a las que hacemos referencia han sido definidas previamente en los datos de configuración estándar del HLR para pruebas funcionales.

Prueba

Desde el VLR del área que está definida como que no soporta Roaming en el HLR vamos a enviar un mensaje de Update Location. Cuando el HLR recibe un Update Location para un abonado desde un VLR situado en un área restringida, el HLR deniega la localización respondiendo con un mensaje de ERROR con causa "Roaming no permitido".

A continuación enviamos otro Update Location desde el VLR que está en el área que sí permite Roaming y esto provoca que el abonado se localice correctamente. Comprobaremos que la prueba ha sido un éxito analizando el mensaje de respuesta del HLR al VLR y comparándolo mediante la plantilla creada en TTCN.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba, restauramos las características de las áreas de los dos VLRs involucrados en la prueba y borramos el abonado por comando, recibiendo el mensaje de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba4 - Update Location – Update Location MAPv2 desde el mismo VLR en el que ya estaba localizado el abonado

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3. Situado en un área que no soporta Roaming.
- **HLR** (SSN = 6) – Nodo levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en los mismos PTCs de TTCN-3 que los VLRs.

Abonado/s:
1 abonado definido durante la prueba en el HLR.

Preparación de la prueba desde el estado inicial

Durante la preparación definimos por comando comando (del operador simulado mediante TTCN-3) un abonado en el HLR, le dotamos del servicio básico de telefonía y le localizamos en el VLR simulado en TTCN utilizando el protocolo MAPv2.

Prueba

La prueba consiste en realizar otro Update Location desde el mismo VLR que en la preparación sobre el mismo abonado que ya tenemos localizado en ese VLR y comprobar que esta se realiza con éxito. Para ello, analizaremos el tráfico de respuesta del HLR al Update Location y al finalizar este, imprimiremos el estado del abonado para ver que está localizado.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba, borramos el abonado involucrado en la prueba por comando, recibiendo el mensaje de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba5 - Update Location – Abonado con estado Purge hace roaming (U.L. MAPv3) a un nuevo VLR

Elementos Involucrados

Nodo/s:

- **VLR1** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **VLR2** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en los mismos PTCs de TTCN-3 que los VLRs.

Abonado/s:
1 abonado definido durante la prueba en el HLR.

Preparación de la prueba desde el estado inicial

Durante la preparación vamos a definir un abonado en el HLR, le damos el servicio básico de telefonía y le localizamos en el VLR1 simulado en TTCN utilizando el protocolo MAPv3.

Desde el mismo VLR1, enviamos un mensaje Purge al abonado localizado, lo cual provoca que el estado del abonado cambie de localizado a purge.

Información adicional: un abonado con el estado *'purged'* se considera como:

'No alcanzable' si el HLR es preguntado por él.
Localización *'desconocido'* para el cambio de datos de abonado.
Localización *'conocida'* si es necesario mandar un Cancel Location hacia el VLR o SGSN.

Prueba

Enviamos un Update Location al HLR para el abonado en estado purge desde el VLR2. Con esta acción desencadenamos dos respuestas del HLR:

1. El HLR cancela la localización en el VLR1 del abonado.
2. El HLR responde afirmativamente al Update Location del VLR2 y queda localizado en él.

Para comprobar que todo se ha realizado correctamente analizamos los mensajes MAP que nos llegan a los PTC's y comprobamos que el estado del abonado es localizado al terminar la prueba (mediante la plantilla TTCN-3).

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba, borramos el abonado involucrado en la prueba por comando (del operador simulado mediante TTCN-3),

recibiendo el mensaje de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba6 - Update Location – – Update Location MAPv3 recibido con error "Dato Inesperado Recibido"

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en los mismos PTCs de TTCN-3 que los VLRs.

Abonado/s:
1 abonado definido durante la prueba en el HLR.

Preparación de la prueba desde el estado inicial

No es necesaria ninguna preparación previa a la ejecución de la prueba.

Prueba

Enviamos un mensaje Update Location con el protocolo MAPv3 desde un VLR con un IMSI de 16 dígitos (el número máximo de dígitos del IMSI es de 15), longitud más larga de lo que marca el estándar.

Comprobamos que se recibe un Error con causa 'Dato Inesperado Recibido'.

Recuperación del estado inicial

No es necesaria ninguna acción para recuperar el estado inicial.

Caso de prueba7 - Update Location – Ejecución Update Location MAPv1 con segmentación del Insert Subscriber Data

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en los mismos PTCs de TTCN-3 que los VLRs.

Abonado/s:
1 abonado definido durante la prueba en el HLR.

Preparación de la prueba desde el estado inicial

Definimos al abonado sobre el que vamos a ejecutar este caso de prueba y le dotamos de suficientes servicios como para que el HLR tenga que segmentar la información en más de un mensaje Insert Subscriber Data.

Vamos a proveer al abonado de los siguientes servicios básicos:

TS11: Telefonía
TS21: Mensaje Corto Terminado en el móvil
BS32: Circuito de datos síncrono de 2400 b/s
BS33: Circuito de datos síncrono de 4800 b/s
BS21: Circuito de datos asíncrono de 300 b/s
BS24: Circuito de datos asíncrono de 2400 b/s
BS25: Circuito de datos asíncrono de 4800 b/s

También vamos a proveer y activar los siguientes servicios:

ICI: IMMEDIATE CALL ITEMIZATION (Desglose de Llamada Inmediato) También llamada 'Hot Billing' se utiliza cuando es necesario tener datos de cobro de salida en tiempo real (por ejemplo para cobrar a una tercera parte)
OIN: ORIGINATING INTELLIGENT NETWORK (Red inteligente con origen en el móvil)
MPTY: MULTIPARTY SERVICE (Servicio Multillamada): Consiste en establecer una conversación entre varios participantes hasta un máximo de 6.
COLP: CONNECTED LINE IDENTIFICATION PRESENTATION (Presentación de identificación de la línea conectada) Esta funcionalidad permite ver el número del abonado que se está conectado a una multiconferencia.
COLR: CONNECTED LINE IDENTIFICATION RESTRICTION (Restricción de identificación de la línea conectada) Esta funcionalidad permite impedir al abonado que está realizando la llamada que su número aparezca al conectarse a una multiconferencia.

CLIP: CALLING LINE IDENTIFICATION PRESENTATION (Presentación de identificación de la línea llamante) Esta funcionalidad permite ver el número del abonado que está realizando la llamada.
CLIR: CALLING LINE IDENTIFICATION RESTRICTION (Restricción de identificación de la línea llamante) Esta funcionalidad permite impedir al abonado que está realizando la llamada que su número aparezca como llamante.
CAW: CALL WAITING (Llamada en espera). Este servicio permite avisar al abonado mientras está manteniendo una conversación de que tiene otra llamada entrante mediante un aviso sonoro. El abonado puede decidir coger la llamada entrante, rechazarla o ignorarla.
HOLD: CALL HOLD (Llamada retenida). Este servicio permite al abonado mantener una llamada retenida mientras está teniendo otra conversación.
AOC: ADVICE OF CHARGE (aviso de facturación): Este servicio da al abonado la información necesaria para calcular el coste de una llamada que va a realizar.
BOIEXH: BARRING OF ALL OUTGOING INTERNATIONAL CALLS EXCEPT THOSE DIRECTED TO THE HOME PLMN COUNTRY (Prohibición de todas las llamadas salientes internacionales excepto aquellas dirigidas a la red fija del país origen)
BOIC: BARRING OF ALL OUTGOING INTERNATIONAL CALLS (Prohibición de todas las llamadas salientes internacionales)
BAOC: BARRING OF ALL OUTGOING CALLS (Prohibición de todas las llamadas salientes)
CFNRY: CALL FORWARDING ON NO REPLY (Desvío de llamada en caso de no tener respuesta)
CFNRC: CALL FORWARDING ON MOBILE SUBSCRIBER NOT REACHABLE (Desvío de llamada en caso de que el abonado destino no esté alcanzable)
CFB: CALL FORWARDING ON MOBILE SUBSCRIBER BUSY (Desvío de llamada en caso de que el abonado destino esté comunicando)
CFU: CALL FORWARDING UNCONDITIONAL (Desvío de llamada incondicional) este servicio permite desviar todas las llamadas entrantes.

Prueba

Realizamos el Update Location y comprobamos que desde el HLR la información del abonado llega segmentada en más de un Insert Subscriber Data. También confirmamos que el abonado queda localizado correctamente comprobando su estado con una plantilla de datos comparada con el resultado devuelto al introducir un comando MML.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba, borramos el abonado involucrado en la prueba, recibiendo el mensaje de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba8 - Update Location – Mensaje Cancel Location devuelto debido a congestión en la red

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en los mismos PTCs de TTCN-3 que los VLRs.

Abonado/s:
1 abonado definido durante la prueba en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso vamos a definir el abonado sobre el que se va a realizar la prueba.

Durante el caso de prueba utilizamos una variable definida en el HLR en la que se almacenan el número de reintentos del mensaje Cancel Location del abonado. La preparación consiste en activar las trazas en el HLR para establecer un valor inicial de esta variable y para ser capaces de comprobar que la variable se ha incrementado después de la prueba.

También vamos a decrementar hasta 1 el número de individuos disponibles para tratar la operación de tráfico Cancel Location en el HLR. Con esto nos aseguramos de que traceando la variable de reintentos de ese individuo, estamos traceando la de nuestro individuo ya que el HLR sólo acepta procesar uno.

Para finalizar la preparación de la prueba localizamos al abonado en el VLR simulado en TTCN.

Prueba

Para simular que hay congestión en la red "tiramos" los enlaces entre el punto de señalización 300 (HLR) y el 410 (VLR) por comando (del operador simulado mediante TTCN-3). Una vez que tenemos el enlace caído, cancelamos la localización del abonado por comando (del operador simulado mediante TTCN-3).

Como resultado de esta cancelación, el HLR debe mandar el mensaje Cancel Location al VLR y esperar su respuesta afirmativa, pero como hemos cortado el enlace, nunca va a recibirla. En este caso el HLR reenvía el mensaje hasta 3 veces (el número de reintentos se puede modificar mediante una variable del HLR) y si no encuentra respuesta libera los recursos reservados para esa operación dándola por finalizada. Podremos verificar que todo ha ido bien con las trazas que hemos activado en la preparación del caso en las que vemos la variable que almacena los reintentos aumentada en 3 (esta comprobación se hace con una plantilla de datos de MML)

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba, primero reestablecemos el enlace entre el HLR y el VLR y a continuación borramos el abonado involucrado en la prueba. En esta ocasión no recibimos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

6.2 Pruebas para la funcionalidad de Update GPRS Location

Caso de prueba 1 - Update GPRS Location – Update GPRS Location sobre un abonado sin acceso a la red de datos.

Elementos Involucrados

Nodo/s:

- **SGSN** (SSN = 149) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el SGSN.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba y le dotamos con el servicio básico de telefonía. Debemos comprobar que el abonado se ha creado con valor 1 en su NAM para poder realizar la prueba correctamente.

Prueba

Utilizando el SGSN creado en TTCN-3 formamos y enviamos un mensaje BEGIN/INVOKE con la operación Update GPRS Location en MAPv3 (es la única versión de MAP que soporta este mensaje).

Este mensaje de actualización de la localización en la red GPRS llega al HLR, el cuál lo analiza y se da cuenta de que el abonado para el cuál llega el mensaje no tiene permitido el acceso a la red GPRS ya que su NAM está a 1, por lo que el HLR (siguiendo la funcionalidad explicada en el capítulo 5.2) debe responder con un mensaje END/ERROR en el que la causa del error será "*Abonado Desconocido*" con valor de diagnóstico 'GPRS'. El PTC que simula el SGSN recibe este mensaje y comprueba que es un mensaje END/ERROR y que la causa es la correcta utilizando una plantilla previamente programado para esta comparación.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibimos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 2 - Update GPRS Location – Congestión en la distribución interna del HLR al recibir un Update GPRS Location.

Elementos Involucrados

Nodo/s:

- **SGSN** (SSN = 149) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el SGSN.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba. Debemos dar al abonado acceso a la red de paquetes poniendo su NAM a 0. Además le dotamos del servicio de mensaje corto, prohibición de las llamadas salientes, y le asignamos un APN. AQUÍ

Información adicional: *APN* (Access Point Name - Nombre del Punto de Acceso): es el nombre de la puerta de acceso entre la red GPRS y otra red (normalmente Internet). En estos casos de tráfico utilizamos APN sin valor real como ERICSSON.MADRID.COM o EXAMPLE.COM.

A continuación decrementamos el valor del SAE 500 del HSDDH a 0 para provocar la congestión.

Información adicional: *SAE* (Size Alteration Event – Evento de Alteración de Tamaño): los SAEs son una parte importante de la Arquitectura AXE, ya que es la forma que se tiene de dimensionar los nodos. El SAE es la reserva de memoria que hacen los bloques software PLEX para almacenar datos (de operaciones y/o abonados principalmente) mientras procesa la operación requerida. Al terminar dicha operación, el SAE ocupado se libera y queda disponible para la siguiente. Evidentemente, si los SAEs se llenan y llega una operación que no tiene "sitio", el HLR mandar congestión como respuesta. En esta prueba simulamos dicho escenario poniendo uno de los SAEs implicados en el Update GPRS Location a 0.

Prueba

Utilizando el SGSN creado en TTCN-3 formamos y enviamos un mensaje BEGIN/INVOKE con la operación Update GPRS Location en MAPv3.

Este mensaje de actualización de la localización en la red GPRS llega al HLR, el cuál detecta que tiene el SAE 500 del HLSDDH a 0, con lo que no puede procesar la operación y devuelve un mensaje ABORT indicando congestión.

El PTC que simula el SGSN recibe este mensaje y comprueba que es un mensaje ABORT y que la causa es la correcta utilizando una plantilla previamente programada para esta comparación.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 3 - Update GPRS Location – HLR cancela la localización tras el cambio de NAM del abonado.

Elementos Involucrados

Nodo/s:

- **SGSN** (SSN = 149) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el SGSN.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba. Debemos dar al abonado el acceso a la red de paquetes poniendo su NAM a 0. Además le dotamos del servicio de mensajes cortos, prohibición de las llamadas salientes, y le asignaremos un APN.

A continuación localizamos al abonado en el dominio de paquetes con un Update GPRS Location enviado desde el SGSN simulado en TTCN con un PTC. El abonado debe localizarse correctamente ya que tiene NAM a 0, es decir, que tiene acceso tanto a la red de circuitos como a la de paquetes.

Prueba

Cambiamos el NAM del abonado de 0 a 1, prohibiendo de este modo el acceso del abonado a la red de paquetes, y permitiéndoselo sólo a la red de circuitos.

El HLR reacciona cancelando la localización del abonado en el SGSN en el cuál lo está mediante el envío de un mensaje Cancel Location. Este mensaje es recibido por el PTC de nuestro caso de prueba, y analizado para comprobar que es correcto.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 4 - Update GPRS Location – Update GPRS Location sobre un IMSI no activo.

Elementos Involucrados

Nodo/s:

- **SGSN** (SSN = 149) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el SGSN.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba. Debemos dar al abonado de acceso a la red de paquetes poniendo su NAM a 0. Además le dotamos del servicio de mensaje corto y realizamos un cambio de IMSI de ejecución inmediata por comando.

Prueba

A continuación localizamos al abonado en el dominio de paquetes con un Update GPRS Location al IMSI no activo (el antiguo IMSI) enviado desde el SGSN simulado en TTCN con un PTC. El HLR detecta que no hay ningún abonado con ese IMSI activo y responde con un mensaje de ERROR con causa “Abonado Desconocido”. El SGS recibe el mensaje de ERROR y lo compara con la plantilla predefinida de TTCN, evidenciando de esta manera que el comportamiento del HLR es correcto.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

6.3 Pruebas para la funcionalidad de IMSI Changeover

Caso de prueba 1 - IMSI Changeover – Cambio de IMSI en estado Pendiente. SIM con IMSI antiguo todavía es aceptada.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR, lo cual es muy importante ya que vamos a programar un cambio de IMSI con fecha futura y debemos controlar perfectamente la fecha actual del sistema.

Introducimos por comando (del operador simulado con TTCN-3) el nuevo IMSI y la fecha en la que se ejecutará ese cambio. Hasta entonces el IMSI antiguo sigue siendo aceptado.

Prueba

Para comprobar que el IMSI antiguo sigue siendo aceptado enviaremos un mensaje de Update Location desde un VLR simulado en TTCN con un PTC sobre el abonado utilizando el IMSI antiguo.

Esto genera un intercambio de mensajes de tráfico entre el PTC y el HLR. Utilizando las plantillas previamente programadas comprobamos que el abonado se localiza correctamente.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba borramos el abonado por comando (del operador simulado con TTCN-3), recibiendo el correspondiente mensaje MAP de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Código TTCN-3 del caso de prueba 1 de IMSI Changeover

Se crea un MTC con la palabra clave *testcase* llamado *Caso1_IMSI_Changeover*, el cuál va a ser del tipo *TCAP_MML_Component*:

Dentro del MTC, únicamente se crea un PTC llamado TCAP_MML_Component_PTC y con los TEST PORTS PARAMETERS definidos para VLR_1_1210 y en ese PTC se inicia la función f_ *Caso1_IMSI_Changeover* que es en la que se va a introducir todo el código de la prueba. Esto se hace así para no utilizar los puertos del MTC y poder ejecutar casos de distintas funcionalidades en la misma ejecución.

Con la instrucción .done se espera a que el PTC acabe la ejecución de la función antes de finalizar el proceso MTC.

```
testcase Caso1_IMSI_Changeover () runs on TCAP_MML_Component
{
  //Variables Declaration.
  var TCAP_MML_Component TCAP_MML_Component_PTC :=
TCAP_MML_Component.create("VLR_1_1210");
  TCAP_MML_Component_PTC.start ( f_Caso1_IMSI_Changeover() );
  TCAP_MML_Component_PTC.done;
}
```

En la función f_Caso_1_Update_Location se va a lanzar la prueba y más PTCs en caso de ser necesarios. Comentar que el tipo del PTC es el mismo que el MTC: TCAP_MML_Component, por lo que posee los mismos puertos que el anterior, pero en este caso sí que se van a utilizar.

```
function f_Caso1_IMSI_Changeover ()runs on TCAP_MML_Component
{
  //Declaración de Variables.
  var boolean v_result := true;
  var ASP_MML_INDICATION v_mml_indication;
  var octetstring v_Msisdn;
  var octetstring v_Imsi;
  var octetstring v_VLR_Number;
  var octetstring v_hlr_number;
  var integer v_numSegments := 1;
  var MAP_Invoke v_mapInvoke_UL;
  var template MAP_Invoke v_mapInvoke_ISD;
  var template MAP_ReturnResult  v_map_Invoke_UL_res;
  var TCAP_PAR_Address v_TcapAddress_Called;
  var TCAP_PAR_Address v_TcapAddress_Calling;
  var object_list v_object :=
  {
    {
      p_typeField := "IMSICHOST",
      p_vals := "2"
    }
  };

  //Convertimos varios variables al tipo adecuado para su uso.
  v_Msisdn := f_Converter_Charstring_To_Map_wPrefix ( px_user_1210.msisdn );
  v_Imsi := f_Converter_Charstring_To_Map ( px_user_1210.imsi );
  v_VLR_Number := f_Converter_Hexstring_To_Map_wPrefix ( px_VLR.number );
  v_hlr_number:= f_Converter_Hexstring_To_Map_wPrefix ( px_HLR.number );
    v_TcapAddress_Called := valueof ( t_TcapAddress ( ( int2bit (
px_nodeDestination_1210.PC,14 ) ), cg_HLR_SSN, px_HLR.number ) );
  v_TcapAddress_Calling := valueof ( t_TcapAddress ( ( int2bit (
px_nodeOrigination_1210.PC,14 ) ), cg_VLR_SSN, px_VLR.number ) );

  //Mapeamos los puertos TCAP y MML contra el SUT.
  map ( self:TCAP_HLR, system:TCAP_HLR );
  map ( self:MML_MTC_PORT, system:MML_MTC_PORT );

  //Se define el abonado en el HLR con el comando HGSUI utilizando los module
parameters definidos en el configuration file para el MSISDN y el IMSI.
```

```
  v_result := f_MML_execute_cmd (
"HGSUI:MSISDN="&px_user_1210.msisdn&",IMSI="&px_user_1210.imsi&";", t_EXECUTED
);
  if ( not v_result ) { goto END; }

  //Se pone en hora el reloj del sistema
  v_result := f_MML_execute_cmd ( "CACLS:DATE=080417,TIME=0800;", t_EXECUTED );

  //Iniciamos el procedimiento de cambio de IMSI con una fecha futura
  v_result := f_MML_execute_cmd (
"HGICI:IMSI="&px_user_1210.imsi&",NIMSI="&px_user2_1210.imsi&",DATE=080418;",
t_EXECUTED );

  if ( not v_result ) { goto END; }

  //Se comprueba que el cambio de IMSI para nuestro abonado está en estado
pendiente
  v_result := f_execute_cmd_withPrintout (
"HGICP:MSISDN="&px_user_1210.msisdn&";",
t_MML_ANSWER_PRINTOUT_SIMPLE_STRING("PEND"), v_mml_indication);
  if ( not v_result ) { goto END; }

//Se crea un INVOKE con el template t_MAP_UpdateLocationInfo_Arg_v2 (MAPv2) al
que se le ha pasado los valores del invoke ID, el imsi y el vlr:

  v_mapInvoke_UL := valueof ( t_MAP_UpdateLocationInfo_Arg_v2 (
px_Invoke_ID_1210, v_Imsi, v_VLR_Number ) );

//Se utiliza la función común f_send_BEGIN_INVOKE para mandar el BEGIN INVOKE
contra el HLR. Se le debe pasar como parámetros el dialogue ID, invoke ID,
application context, dirección de destino y de origen, código de la operación
MAP del Update Location, el rango de clase, el invoke que he construido antes y
true  o false dependiendo si quiero que se codifique el invoke o no. Remarcar
que esta función ya hace uso directamente de los puertos TCAP del PTC con la
instrucción .send:

  f_send_BEGIN_INVOKE ( px_Dialogue_ID_1210, px_Invoke_ID_1210,
cg_MAP_applicationContext_UpdateLocation_V2, v_TcapAddress_Called,
    v_TcapAddress_Calling, cg_MAP_operationCode_UpdateLocation,
succAndFailReported, v_mapInvoke_UL, true );

//Ahora se debe recibir el mensaje Insert Subscriber Data: CONTINUE/INVOKE
procedente del HLR y enviar el CONTINUE/RESULT. Para ello se hace uso de otra
función común, f_receive_CONTINUEinv_send_CONTINUEresl, a la que se le pasa como
parámetros el dialogue ID, application contextname, código de la operación,
invoke ID, true o false para pedir que decodifique el invoke, un template si
queremos chequear en invoke recibido y un timer con el tiempo de espera máximo.

  //Receive: CONTINUE + INVOKE (InsertSubscriberData).
  //Send: CONTINUE + RESULT_L (InsertSubscriberData EMPTY).
  v_mapInvoke_ISD := t_MAP_InsertSubscriberData_Arg_v2( v_Msisdn);
  v_result := f_receive_CONTINUEinv_send_CONTINUEresl ( px_Dialogue_ID_1210,
cg_MAP_applicationContext_UpdateLocation_V2,
    cg_MAP_operationCode_InsertSD, v_mapInvoke_ISD, true, omit, false,
v_numSegments, px_Timer_Value_1210 );

//Si no ha ido bien la recepción, se loguea el error en la función y se salta a
la etiqueta END para finalizar el caso de prueba.

  if ( not v_result )
  {
    log( "Error: f_receive_CONTINUEinv_send_CONTINUEresl: InsertSubscriberData"
);
    goto END;
  }
```

```
//Se recibe el mensaje END/RESULT del Update Location MAPv2 y si no va bien se
loguea el error y se salta a la etiqueta END

  v_result := f_receive_END_RESULT_L ( px_Dialogue_ID_1210, px_Invoke_ID_1210,
cg_MAP_applicationContext_UpdateLocation_V2,
    cg_MAP_operationCode_UpdateLocation, ?, false, px_Timer_Value_1210 );
  if ( not v_result )
  {
    log( "Error: f_receive_END_RESULT_L: UpdateLocation" );
    goto END;
  }

//Se comprueba que el cambio de IMSI sigue en estado pendiente ya que se ha
realizado el Update Location con el IMSI antiguo y por lo tanto no se ha
efectuado el cambio.

    v_result := f_execute_cmd_withPrintout (
"HGICP:MSISDN="&px_user_1210.msisdn&";",
t_MML_ANSWER_PRINTOUT_SIMPLE_STRING("PEND"), v_mml_indication);

  if ( not v_result ) { goto END; }

  label END;

//Se utiliza la función común para borrar el abonado y esperar el Cancel
Location

  v_result := f_DeleteUser_CancelLocation ( px_user_1210.msisdn, v_Imsi,
px_Timer_Value_1210,  px_monolithic);
  if ( not v_result )
  {
    log ( "Error: f_DeleteUser_CancelLocation: DeleteSubscriberData" );
  }

  //Desmapeamos los puertos TCAP y MML
  unmap ( self:TCAP_HLR, system:TCAP_HLR );
  unmap ( self:MML_MTC_PORT, system:MML_MTC_PORT );
```

Caso de prueba 2 - IMSI Changeover – Cambio de IMSI en estado Forzado. SIM con IMSI antiguo es rechazada.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR

Introducimos por comando del operador simulado en TTCN-3 el nuevo IMSI sin fecha de cambio, lo que significa que el cambio se ejecuta inmediatamente y el cambio de IMSI queda en estado Forzado. A partir de ese momento el antiguo IMISI no es aceptado.

Prueba

Para comprobarlo enviamos un mensaje de Update Location desde un VLR simulado en TTCN con un PTC sobre el abonado utilizando el IMSI antiguo.

Debemos recibir un mensaje de ERROR desde el HLR con la razón de "Abonado Desconocido". Utilizando las plantillas previamente programadas certificaremos que el mensaje recibido es el correcto.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 3 - IMSI Changeover – Cambio de IMSI iniciado con la fecha actual. El cambio se ejecuta y queda en estado Forzado.

Elementos Involucrados

Nodo/s:

- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR.

Prueba

Introducimos por comando del operador simulado en TTCN-3 el nuevo IMSI con fecha de cambio la fecha actual, lo que significa que el cambio se ejecuta inmediatamente y el cambio de IMSI queda en estado Forzado. A partir de ese momento el antiguo IMISI no es aceptado.

Para comprobar que la prueba ha sido exitosa, imprimimos mediante comando del operador simulado con TTCN-3 el estado de los datos de IMSI Changeover del abonado, verificando mediante TTCN-3 que está en estado Forzado.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 4 - IMSI Changeover – Cambio de IMSI iniciado con fecha futura. El cambio se marca Pendiente hasta esa fecha aunque se cambie la fecha futura.

Elementos Involucrados

Nodo/s:

- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR.

Prueba

Introducimos por comando el nuevo IMSI con fecha de cambio una fecha futura, lo que significa que el cambio queda en estado Pendiente. Para comprobarlo, imprimimos por comando del operador simulado con TTCN-3 el estado de los datos de IMSI Changeover del abonado, viendo (comprobando en la plantilla TTCN-3) que, efectivamente, está en estado Pendiente.

A continuación cambiaremos la fecha en la que está programado el cambio de IMSI a una fecha más futura que la anterior. Para verificar que la prueba ha sido exitosa, imprimimos por comando del operador simulado con TTCN-3 el estado de los datos de IMSI Changeover del abonado, viendo que sigue en estado Pendiente.

Como fin de la prueba, cambiamos la fecha del sistema a una fecha posterior a la última programada para el cambio de IMSI y realizamos un procedimiento de abonado por comando MML del operador simulado con TTCN-3 con lo que se dispara el cambio.

Para comprobar que la prueba ha sido exitosa, imprimimos por comando del operador simulado con TTCN-3 el estado de los datos de IMSI Changeover del abonado, viendo que está en estado Ejecutado.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 5 - IMSI Changeover – Abortar por comando un cambio de IMSI en estado Pendiente.

Elementos Involucrados

Nodo/s:

- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR.

Prueba

Introducimos por comando del operador simulado con TTCN-3 el nuevo IMSI con fecha de cambio una fecha futura, lo que significa que el cambio queda el estado Pendiente. Para comprobarlo, imprimimos por comando del operador simulado con TTCN-3 el estado de los datos de IMSI Changeover del abonado, viendo que está en estado Pendiente.

Abortamos el cambio de IMSI por comando del operador simulado con TTCN-3.

Para comprobar que la prueba ha sido exitosa, imprimimos por comando del operador simulado con TTCN-3 el estado de los datos de IMSI Changeover del abonado, constatando que no hay ningún cambio pendiente.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 6 - IMSI Changeover – Cambio de IMSI en estado Pendiente. Se ejecuta al recibir el mensaje ready for SM.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR

Introducimos por comando del operador simulado mediante TTCN-3 el nuevo IMSI con fecha futura, lo que significa que el cambio queda en estado Pendiente.

Prueba

Para ejecutar el cambio enviamos un mensaje Ready for SM desde un VLR simulado en TTCN con un PTC sobre el abonado utilizando el IMSI nuevo.

Este mensaje desencadena el cambio de IMSI.

Para comprobar que la prueba ha sido exitosa, imprimimos por comando del operador simulado con TTCN-3 el estado de los datos de IMSI Changeover del abonado, viendo que el estado es forzado.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 7 - IMSI Changeover – Update GPRS Loc. Recibido para nuevo IMSI de un abonado localizado en red GPRS y non-GPRS.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR.

Introducimos por comando del operador simulado con TTCN-3 el nuevo IMSI con fecha futura, lo que significa que el cambio queda en estado Pendiente. Localizamos el abonado con el IMSI antiguo en la red GPRS con un Update GPRS Location desde un SGSN simulado en TTCN con un PTC. Localizamos el abonado con el IMSI antiguo en la red non-GPRS con un Update Location desde un VLR simulado en TTCN con un PTC.

Preparamos en TTCN los PTCs para recibir los mensajes de Cancel Location de ambas redes.

Prueba

Para ejecutar el cambio de IMSI, enviamos un Update GPRS Location desde un SGSN simulado en TTCN con un PTC sobre el abonado utilizando el IMSI nuevo.

Como consecuencia el HLR envía dos Cancel Location sobre el SGSN y VLR en los que estaba localizado el abonado con el IMSI antiguo y completa la localización en el SGN desde el que se ha pedido con el nuevo IMSI, quedando el cambio de IMSI en estado Forzado.

Para comprobar que la prueba ha sido exitosa, imprimimos por comando del operador simulado mediante TTCN el estado de los datos de IMSI Changeover del abonado, verificando que el estado es Forzado.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba borramos el abonado por comando, recibiendo el correspondiente mensaje MAP de Cancel Location desde el HLR

al SGNS para informar al SGSN que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba 8 - IMSI Changeover – Eliminación del IMSI pasivo una vez que el cambio de IMSI se ha producido.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba. A continuación ponemos en fecha y hora el reloj del HLR.

Tenemos que introducir la fecha del sistema y cambiar el NAM (Modo de Acceso a la Red) a 2 para que tenga acceso a la red GPRS. Vamos a dotar al abonado del servicio básico de telefonía y realizamos un Update GPRS Location sobre el abonado definido desde un SGSN simulado con un PTC en TTCN.

Introducimos por comando del operador simulado mediante TTCN el nuevo IMSI con fecha futura, lo que significa que el cambio queda en estado Pendiente.

Generamos un mensaje MAP avisando que el abonado está listo para recibir un mensaje corto desde el PTC de TTCN que simula el VLR, con el nuevo IMSI y comprobamos que el cambio de IMSI se ha producido imprimiendo los datos del abonado, comprobando que el estado ha quedado en Ejecutado. También recibimos un mensaje de Cancelación de la Localización del HLR hacia el SGSN sobre el que el abonado estaba localizado en la red GPRS.

Prueba

Eliminamos toda la información referente al IMSI antiguo por medio de comando.

Para comprobar que esto se ha realizado, imprimimos la información del abonado de nuevo comprobando si hay alguna referencia al IMSI antiguo.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

6.4 Pruebas para la funcionalidad de Rechazo de Llamada Anónima

Caso de prueba 1 – ACR – Código USSD con formato inválido.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba. A continuación realizamos la localización del abonado en la red non-GPRS con una operación Update Location MAPv1 desde un PTC en TTCN.

Prueba

Mandamos desde el VLR una operación Begin Subscriber Activity con el código USSD incorrecto ##254*#, la cuál es respondida desde el HLR como Operación No Reconocida.

Con el intercambio de mensajes de tráfico entre el PTC y el HLR y utilizando las plantillas previamente programadas en TTCN comprobaremos que el intercambio de mensajes es correcto.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba borramos el abonado por comando del operador simulado mediante TTCN, recibiendo el correspondiente mensaje MAP de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba 2 – ACR – Servicio ACR no provisto al abonado.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba. A continuación realizamos la localización del abonado en la red non-GPRS con una operación Update Location MAPv1 desde un PTC en TTCN.

Prueba

Mandamos desde el VLR una operación Begin Subscriber Activity con el código USSD de desactivación del servicio ACR #254*#, la cuál es respondida desde el HLR como Servicio No Disponible.

Con el intercambio de mensajes de tráfico entre el PTC y el HLR y utilizando las plantillas previamente programadas en TTCN comprobamos que el intercambio de mensajes es correcto.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba borramos el abonado por comando del operador simulado con TTCN-3, recibiendo el correspondiente mensaje MAP de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba 3 – ACR – Activación del servicio ACR.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso vamos a definir el abonado sobre el que se va a realizar la prueba y proveerle del servicio ACR por comando. A continuación vamos realizar la localización del abonado en la red non-GPRS con una operación Update Location MAPv1 desde un PTC en TTCN.

Prueba

La prueba consiste en mandar desde el VLR una operación Begin Subscriber Activity con el código USSD de activación del servicio ACR *254*#, la cual provoca que el HLR marque el servicio ACR como activo entre los datos de este abonado y responda diciendo que la operación ha sido un éxito.

Con el intercambio de mensajes de tráfico entre el PTC y el HLR y las plantillas previamente programadas en TTCN comprobamos que el intercambio de mensajes es correcto. También imprimimos los datos del abonado comprobando que tiene el servicio ACR con el valor 2, que significa activado.

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba borramos el abonado por comando del operador simulado mediante TTCN, recibiendo el correspondiente mensaje MAP de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba 4 – ACR – Desactivación del servicio ACR.

Elementos Involucrados

Nodo/s:

- **VLR** (SSN = 7) – Simulado en un PTC de TTCN-3.
- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba y le proveemos del servicio ACR ya activado por comando. A continuación vamos realizar la localización del abonado en la red non-GPRS con una operación Update Location MAPv1 desde un PTC en TTCN.

Prueba

Mandamos desde el VLR una operación Begin Subscriber Activity con el código USSD de desactivación del servicio ACR #254*#, la cual hace que el HLR marque el servicio ACR como desactivado para ese abonado y responda como Operación Exitosa.

Con el intercambio de mensajes de tráfico entre el PTC y el HLR y usando las plantillas previamente programadas comprobamos que el intercambio de mensajes es correcto. También imprimimos los datos del abonado y comprobamos que el servicio ACR tiene el valor 1 (desactivo).

Recuperación del estado inicial

Para dejar el HLR como estaba antes de nuestra prueba borramos el abonado por comando del operador simulado con TTCN-3, recibiendo el correspondiente mensaje MAP de Cancel Location desde el HLR al VLR para informar al VLR que dicho abonado ya no debe figurar como localizado en sus datos.

Caso de prueba 5 – ACR – Provisión, activación, desactivación y retirada del servicio ACR.

Elementos Involucrados

Nodo/s:

- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba.

Prueba

Por comando MML, ponemos el servicio ACR del abonado en todos los estados posibles: Provisto (1) → Activo (2) → Provisto (1) → Retirado (0) → Activo (2) → Retirado (0).

Comprobamos que todos los resultados de los comandos son exitosos con las correspondientes plantillas pre programadas en TTCN.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 6 – ACR – Retirada del servicio por cambio de dominio.

Elementos Involucrados

Nodo/s:

- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso creamos el abonado sobre el que se va a realizar la prueba.

Prueba

Por comando MML del operador simulado con TTCN-3, ponemos el servicio ACR del abonado en estado activo. A continuación cambiamos el NAM del abonado a sólo GPRS (NAM=2) sin introducir el parámetro KEEP, esto provoca que el servicio ACR pase a estado retirado para el abonado.

Comprobamos el éxito de la prueba imprimiendo los datos del abonado y chequeando que el valor del servicio ACR es 0.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Caso de prueba 7 – ACR – No retirada del servicio, a pesar de cambio de dominio.

Elementos Involucrados

Nodo/s:

- **HLR** (SSN = 6) – Nodo bajo prueba levantado utilizando la herramienta SEA.
- **Operador** – Simula a un administrador de red que envía comandos MML al HLR para configurar dicho nodo. Está simulado en el mismo PTC de TTCN-3 que el VLR.

Abonado/s:
1 abonado definido en el HLR.

Preparación de la prueba desde el estado inicial

En la preparación de este caso definimos el abonado sobre el que se va a realizar la prueba.

Prueba

Por comando MML del operador simulado con TTCN-3, ponemos el servicio ACR del abonado en estado activo. A continuación cambiamos el NAM del abonado a sólo GPRS (NAM=2) introduciendo el parámetro KEEP; esto provoca que el servicio ACR continúe en el mismo estado para el abonado.

Comprobamos el éxito de la prueba imprimiendo los datos del abonado y verificando que el valor del servicio ACR continúa siendo 2.

Recuperación del estado inicial

Para recuperar el estado inicial borramos el abonado definido durante la preparación. En esta ocasión no recibiremos mensaje de Cancel Location desde el HLR ya que no estaba localizado en el momento del borrado.

Capítulo 7

CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

7.1 Conclusiones

Durante este proyecto hemos podido comprobar cómo la automatización de las pruebas funcionales, combinada con una nueva metodología de procesos como el desarrollo ágil del software puede hacer mejorar los tiempos de desarrollo y la calidad del producto.

El uso de lenguajes orientados a pruebas como el TTCN-3, nos da la posibilidad de programar pruebas que pueden ser ejecutadas en cualquier momento en el tiempo para comprobar que la calidad del producto se mantiene. Durante el proyecto se ha comprobado que estas pruebas deben seguir ciertas reglas a la hora de ser construidas, ya que de lo contrario serían poco eficaces y la calidad final del producto se resentiría; las pruebas deben tener el foco en probar que la funcionalidad del producto sea la correcta, no en conseguir pasar la prueba en sí misma. Esto puede parecer algo obvio aunque en entornos profesionales con gran cantidad de pruebas por realizar y mucha presión de tiempo no lo es tanto. Otra recomendación es que las pruebas deben ser lo más generales posibles, es decir, no programar la prueba dependiente del entorno, de los datos del producto o de la configuración. Las pruebas también deben ser robustas, aunque cuanto más robustas las queramos más tiempo deberemos invertir, aquí nos encontramos con la

difícil posición de encontrar el punto exacto de compromiso entre calidad de la prueba y tiempo invertido en ella.

Otra importante conclusión del proyecto es que el entorno utilizado durante las pruebas debe ser fiable. Muchas veces el entorno de pruebas se ve obligado a evolucionar con el producto, a adaptarse a él y si no se hace correctamente durante las pruebas se puede perder mucho tiempo. Por ejemplo, en el caso del proyecto si las funciones comunes utilizadas no funcionan correctamente, el probador se pasará mucho tiempo encontrando el fallo en esas funciones en vez de encontrando el fallo en el producto que es lo que realmente aporta valor añadido.

¿Se deben automatizar todas las pruebas? Tras la ejecución de este proyecto podemos decir que no. No todas las pruebas pueden ni deben ser automatizadas, hay pruebas que técnicamente no es posible automatizar y otras que por la dificultad que presenta su automatización o por que no son susceptibles de ser repetidas en el tiempo, no merece la pena invertir recursos en su automatización.

Una vez una prueba es automatizada se puede ejecutar N veces, pero hay que tener en cuanta que siempre tendrá un coste de mantenimiento ya que hace uso de un entorno que necesita ser cuidado, actualizado y evolucionado. Incluso las pruebas pueden tener que ser actualizadas debido a algún cambio profundo del entorno de pruebas.

7.2 Líneas de trabajo futuras

Las pruebas realizadas al nuevo software comercial y su automatización son uno de los caballos de batalla en las empresas actuales ya que suponen una gran inversión que debe ser ajustada para obtener la calidad requerida en el producto y llegar al mercado en el momento deseado. Un producto de gran calidad pero que llegue tarde al mercado y a un precio elevado puede no ser rentable para la compañía.

Teniendo esto en cuenta las posibilidades de trabajos futuros en este campo son inmensas. Se pueden proponer estudios de como diferentes empresas utilizan la automatización de pruebas y los resultados obtenidos, estudios sobre el compromiso entre la calidad del producto y la inversión en pruebas, introduciendo la variable “time to market” (tiempo que tarda el producto en estar en el mercado).

Otra línea de trabajos futuro más técnica puede estar relacionada con utilizar el lenguaje TTCN-3 sobre otro producto comercial distinto de un nodo de telecomunicaciones y hacer una comparativa.

La ampliación de las pruebas del HLR sería algo muy interesante, ampliar este proyecto al nuevo producto HLR-FE que posee un entorno mucho más complejo con más herramientas involucradas.

En este proyecto se ha trabajado sobre las pruebas funcionales, pero en otro futuro proyecto se podría tratar las pruebas de carga del producto, realizadas con TTCN-3 u otro generador de tráfico comercial.

Glosario

ACR	*Anonymous Call Rejection*
ANSI	*American National Standards Institute*
APN	*Access Point Name*
AUC	*Authentication Centre*
AXE	*Automatic Cross-Connection Equipment*
FNR	*Flexible Number Register*
GMSC	*Gateway Mobile Switching Centre*
GSM	*Global System for Mobile Communications*
GPRS	*General Packet Radio Service*
gsmSCF	*GSM Service Control Function*
gsmSSF	*GSM Service Switching Function*
HLR	*Home Location Register*
IMSI	*International Mobile Subscriber Identity*
IN	*Intelligent Network*
ITU	*International Telecommunication Union*
MAP	*Mobile Application Part*
MML	*Man Machine Language*
MSC	*Mobile Switching Centre*
MSISDN	*Mobile Subscriber ISDN [Integrated Services Digital Network] Number*
MSRN	*Mobile Station Roaming Number*
MTC	*Main Test Component*
NAM	*Network Access Mode*
OSS	*Operation Support System*
PTC	*Parallel Test Component*
RDSI	*Red Digital de Servicios Integrados*
SAPC	*Service-Aware Policy Controller*
SASN	*Service-Aware Support Node*
SCP	*Service Control Point*
SGSN	*Serving GPRS Support Node*
SIM	*Subscriber Identity Module*
SMS-GMSC	*Short Message Service GMSC*
SMS-IWGMSC	*SMS InterWorking GMSC*
SOG	*Service Order Gateway*
SS7	*Sistema de Señalización número 7*
SUT	*System Under Test*
TTCN-3	*Testing and Test Control Notation version 3*
UMTS	*Universal Mobile Telecommunications System - 3ªGeneración (3G)*
VLR	*Visitor Location Register*
WCDMA	*Wideband Code Division Multiple Access – 3rd*

Generation (3G)

ANEXO I: LOG

Logs de la ejecución de pruebas

En la ejecución de pruebas con TTCN-3 se crea un fichero de log para el MTC y otro para cada PTC creado con la nomenclatura: Nombre_Binario.server-NúmPTC.log, por ejemplo para el MTC y el primer PTC creados serían:

HLR-FW-1-9.gtttcnss7v2-mtc.log
HLR-FW-1-9.gtttcnss7v2-3.log

En ciertas pruebas conviene también, además de tener estos logs, lanzar alguna herramienta complementaria como un analizador de protocolos, en este proyecto se ha utilizado el Wireshark, software libre y adaptable a cualquier plataforma.

A continuación se muestran los logs del primer caso de prueba del Update Location. He marcado en rojo las partes que son interesantes para el lector y puede relacionar con las partes explicadas anteriormente y en azul alguna explicación extra introducida:

```
2013/Sep/28 18:21:20.619049 - TTCN-3 Parallel Test Component started on gtttcnss7v2.
Component reference: VLR_1_1106(3), component type:
ComponentDefinitions.TCAP_MML_Component, component name: VLR_1_1106. Version: CRL 113
200/3 R1A.
2013/Sep/28 18:21:20.619155 - TTCN Logger v2.2 options: TimeStampFormat:=DateTime;
LogEntityName:=No; LogEventTypes:=No; SourceInfoFormat:=Single; *.FileMask:=LOG_ALL |
MATCHING; *.ConsoleMask:=ERROR | TESTCASE | STATISTICS; LogFileSize:=0; LogFileNumber:=1;
DiskFullAction:=Error
2013/Sep/28 18:21:20.619295 - Connected to MC.
2013/Sep/28 18:21:20.619324 - Initializing variables, timers and ports of component type
ComponentDefinitions.TCAP_MML_Component inside testcase Caso_1_Update_Location.
2013/Sep/28 18:21:20.619814 - Port LDAP_OpenLDAP was started.
2013/Sep/28 18:21:20.619843 - Port LDAP_HLR was started.
2013/Sep/28 18:21:20.619857 - Port SoapLibraryHTTP was started.
2013/Sep/28 18:21:20.619869 - Port TCAP_HLR was started.
2013/Sep/28 18:21:20.619882 - Port MML_MTC_PORT was started.
2013/Sep/28 18:21:20.619894 - Port MML_MTC_PORT2 was started.
2013/Sep/28 18:21:20.619904 - Port MML_MTC_PORT3 was started.
2013/Sep/28 18:21:20.619915 - Port MML_MTC_PORT4 was started.
2013/Sep/28 18:21:20.619926 - Port MML_MTC_PORT5 was started.
2013/Sep/28 18:21:20.619937 - Port MML_MTC_PORT6 was started.
2013/Sep/28 18:21:20.619947 - Port MML_MTC_PORT7 was started.
2013/Sep/28 18:21:20.619958 - Port MML_MTC_PORT8 was started.
2013/Sep/28 18:21:20.620220 - Component type ComponentDefinitions.TCAP_MML_Component was
initialized.
2013/Sep/28 18:21:20.620400 - Starting function f_Caso_1_Update_Location().
2013/Sep/28 18:21:20.620435 TD1106_upd_loc.ttcn:575 Mapping port VLR_1_1106(3):TCAP_HLR to
system:TCAP_HLR.
2013/Sep/28 18:21:20.780062 TD1106_upd_loc.ttcn:575 Port TCAP_HLR was mapped to
system:TCAP_HLR.
2013/Sep/28 18:21:20.780186 TD1106_upd_loc.ttcn:575 Map operation of
VLR_1_1106(3):TCAP_HLR to system:TCAP_HLR finished.
2013/Sep/28 18:21:20.780213 TD1106_upd_loc.ttcn:576 Mapping port
VLR_1_1106(3):MML_MTC_PORT to system:MML_MTC_PORT.
```

```
2013/Sep/28 18:21:20.780274 TD1106_upd_loc.ttcn:576 MML_MTC_PORT: set_parameter (Hostname
= 10.43.55.200)
2013/Sep/28 18:21:20.780294 TD1106_upd_loc.ttcn:576 MML_MTC_PORT: set_parameter (HttpPort
= 3001)
2013/Sep/28 18:21:20.780309 TD1106_upd_loc.ttcn:576 MML_MTC_PORT: set_parameter
(TerminalName = AD-5)
2013/Sep/28 18:21:20.780323 TD1106_upd_loc.ttcn:576 MML_MTC_PORT: set_parameter
(SeparatedProcedurePrintout = ON)
2013/Sep/28 18:21:20.780338 TD1106_upd_loc.ttcn:576 MML_MTC_PORT: set_parameter
(AnswerPrintout_in_RecordOf = ON)
2013/Sep/28 18:21:20.783331 TD1106_upd_loc.ttcn:576 Port MML_MTC_PORT was mapped to
system:MML_MTC_PORT.
2013/Sep/28 18:21:20.783498 TD1106_upd_loc.ttcn:576 Map operation of
VLR_1_1106(3):MML_MTC_PORT to system:MML_MTC_PORT finished.
```

//Aquí comienzan los comandos de preparación del HLR incluídas en la función f_Manual_Preparations();

```
2013/Sep/28 18:21:20.784204 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"HGEPC:PROP=AUTHDMAND-1;"
2013/Sep/28 18:21:20.784289 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:20.784373 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "HGEPC", commandParameters := "PROP=AUTHDMAND-1" } } }
2013/Sep/28 18:21:20.873779 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := EXECUTED (0), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 1
2013/Sep/28 18:21:20.873846 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:20.873875 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := EXECUTED (0),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 1
2013/Sep/28 18:21:20.873899 MML_Functions.ttcn:994 Message with id 1 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:20.873921 MML_Functions.ttcn:998 setverdict(pass): none -> pass
2013/Sep/28 18:21:20.873949 MML_Functions.ttcn:1001 MML: Command OK, printout: EXECUTED
(0)
2013/Sep/28 18:21:20.874205 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"HGEPC:PROP=MV2RESTRICTION-2;"
2013/Sep/28 18:21:20.874256 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:20.874289 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "HGEPC", commandParameters := "PROP=MV2RESTRICTION-2" } } }
2013/Sep/28 18:21:20.965742 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := EXECUTED (0), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 2
2013/Sep/28 18:21:20.965791 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:20.965813 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := EXECUTED (0),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 2
2013/Sep/28 18:21:20.965833 MML_Functions.ttcn:994 Message with id 2 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:20.965849 MML_Functions.ttcn:998 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:20.965865 MML_Functions.ttcn:1001 MML: Command OK, printout: EXECUTED
(0)
2013/Sep/28 18:21:20.966116 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"HGEPC:PROP=MV3RESTRICTION-2;"
2013/Sep/28 18:21:20.966166 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:20.966198 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "HGEPC", commandParameters := "PROP=MV3RESTRICTION-2" } } }
2013/Sep/28 18:21:21.061705 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := EXECUTED (0), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 3
2013/Sep/28 18:21:21.061760 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:21.061783 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
```

```
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := EXECUTED (0),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 3
2013/Sep/28 18:21:21.061803 MML_Functions.ttcn:994 Message with id 3 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:21.061820 MML_Functions.ttcn:998 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:21.061836 MML_Functions.ttcn:1001 MML: Command OK, printout: EXECUTED
(0)
2013/Sep/28 18:21:21.062087 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"HGEPC:PROP=AUTCHOVERMAND-1;"
2013/Sep/28 18:21:21.062138 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:21.062170 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "HGEPC", commandParameters := "PROP=AUTCHOVERMAND-1" } } }
2013/Sep/28 18:21:21.153767 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := EXECUTED (0), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 4
2013/Sep/28 18:21:21.153815 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:21.153837 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := EXECUTED (0),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 4
2013/Sep/28 18:21:21.153857 MML_Functions.ttcn:994 Message with id 4 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:21.153873 MML_Functions.ttcn:998 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:21.153889 MML_Functions.ttcn:1001 MML: Command OK, printout: EXECUTED
(0)
2013/Sep/28 18:21:21.154090 MML_Functions.ttcn:606 MML: Command is NOT a PG CLI command:
"SAAEP:BLOCK=HISDAP3,SAE=500;"
2013/Sep/28 18:21:21.154339 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"SAAEP:BLOCK=HISDAP3,SAE=500;"
2013/Sep/28 18:21:21.154392 MML_Functions.ttcn:1604 Start timer t_timer: 15 s
2013/Sep/28 18:21:21.154424 MML_Functions.ttcn:1608 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "SAAEP", commandParameters := "BLOCK=HISDAP3,SAE=500" } } }
2013/Sep/28 18:21:21.205720 MML_Functions.ttcn:1610 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0), parameters := { printout
:= { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES INFORMATION", "", "SAE
BLOCK    CNTRTYP  NI          NIU         NIE         NIR", " 500   HISDAP3  CONS1
10000", "END" } } } } id 5
2013/Sep/28 18:21:21.206593 MML_Functions.ttcn:1612 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:21.206631 MML_Functions.ttcn:1612 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0),
parameters := { printout := { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES
INFORMATION", "", "SAE    BLOCK    CNTRTYP  NI          NIU         NIE         NIR", "
500   HISDAP3  CONS1         10000", "END" } } } } id 5
2013/Sep/28 18:21:21.206658 MML_Functions.ttcn:1612 Message with id 5 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:21.206714 MML_Functions.ttcn:1616 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:21.206733 MML_Functions.ttcn:1619 MML: Printout match
2013/Sep/28 18:21:21.206746 MML_Functions.ttcn:1620 Stop timer t_timer: 15 s
2013/Sep/28 18:21:21.207015 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"SAADI:BLOCK=HISDAP3,SAE=500,NI=0;"
2013/Sep/28 18:21:21.207068 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:21.207098 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "SAADI", commandParameters := "BLOCK=HISDAP3,SAE=500,NI=0" } } }
2013/Sep/28 18:21:21.301647 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := ORDERED (1), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 6
2013/Sep/28 18:21:21.301712 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:21.301735 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := ORDERED (1),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 6
2013/Sep/28 18:21:21.301755 MML_Functions.ttcn:994 Message with id 6 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:21.301771 MML_Functions.ttcn:998 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:21.301787 MML_Functions.ttcn:1001 MML: Command OK, printout: ORDERED (1)
```

```
2013/Sep/28 18:21:21.301806 TD1106_upd_loc.ttcn:286 Start timer tl_guard1: 200 s
2013/Sep/28 18:21:22.078168 TD1106_upd_loc.ttcn:287 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0), parameters := { printout
:= { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES RESULT", "", "SAE    BLOCK
NI          NIE         NIR", " 500   HISDAP3           0", "END" } } } } id 7
2013/Sep/28 18:21:22.078818 TD1106_upd_loc.ttcn:289 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:22.078855 TD1106_upd_loc.ttcn:289 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0),
parameters := { printout := { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES
RESULT", "", "SAE    BLOCK    NI          NIE         NIR", " 500   HISDAP3           0",
"END" } } } } id 7
2013/Sep/28 18:21:22.078877 TD1106_upd_loc.ttcn:289 Message with id 7 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:22.078925 TD1106_upd_loc.ttcn:291 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:22.078943 TD1106_upd_loc.ttcn:292 Stop timer tl_guard1: 200 s
2013/Sep/28 18:21:22.079158 MML_Functions.ttcn:606 MML: Command is NOT a PG CLI command:
"SAAEP:BLOCK=HDSDAP3,SAE=500;"
2013/Sep/28 18:21:22.079391 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"SAAEP:BLOCK=HDSDAP3,SAE=500;"
2013/Sep/28 18:21:22.079441 MML_Functions.ttcn:1604 Start timer t_timer: 15 s
2013/Sep/28 18:21:22.079474 MML_Functions.ttcn:1608 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "SAAEP", commandParameters := "BLOCK=HDSDAP3,SAE=500" } } }
2013/Sep/28 18:21:22.125513 MML_Functions.ttcn:1610 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0), parameters := { printout
:= { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES INFORMATION", "", "SAE
BLOCK    CNTRTYP  NI          NIU         NIE         NIR", " 500   HDSDAP3  CONS1
10000", "END" } } } } id 8
2013/Sep/28 18:21:22.126334 MML_Functions.ttcn:1612 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:22.126373 MML_Functions.ttcn:1612 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0),
parameters := { printout := { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES
INFORMATION", "", "SAE    BLOCK    CNTRTYP  NI          NIU         NIE         NIR", "
500   HDSDAP3  CONS1         10000", "END" } } } } id 8
2013/Sep/28 18:21:22.126401 MML_Functions.ttcn:1612 Message with id 8 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:22.126473 MML_Functions.ttcn:1616 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:22.126491 MML_Functions.ttcn:1619 MML: Printout match
2013/Sep/28 18:21:22.126504 MML_Functions.ttcn:1620 Stop timer t_timer: 15 s
2013/Sep/28 18:21:22.126751 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"SAADI:BLOCK=HDSDAP3,SAE=500,NI=0;"
2013/Sep/28 18:21:22.126802 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:22.126833 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "SAADI", commandParameters := "BLOCK=HDSDAP3,SAE=500,NI=0" } } }
2013/Sep/28 18:21:22.221352 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := ORDERED (1), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 9
2013/Sep/28 18:21:22.221420 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:22.221443 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := ORDERED (1),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 9
2013/Sep/28 18:21:22.221463 MML_Functions.ttcn:994 Message with id 9 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:22.221480 MML_Functions.ttcn:998 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:22.221496 MML_Functions.ttcn:1001 MML: Command OK, printout: ORDERED (1)
2013/Sep/28 18:21:22.221515 TD1106_upd_loc.ttcn:320 Start timer tl_guard2: 200 s
2013/Sep/28 18:21:22.989707 TD1106_upd_loc.ttcn:321 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0), parameters := { printout
:= { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES RESULT", "", "SAE    BLOCK
NI          NIE         NIR", " 500   HDSDAP3           0", "END" } } } } id 10
2013/Sep/28 18:21:22.990401 TD1106_upd_loc.ttcn:323 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:22.990439 TD1106_upd_loc.ttcn:323 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0),
parameters := { printout := { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES
RESULT", "", "SAE    BLOCK    NI          NIE         NIR", " 500   HDSDAP3           0",
"END" } } } } id 10
```

```
2013/Sep/28 18:21:22.990461 TD1106_upd_loc.ttcn:323 Message with id 10 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:22.990509 TD1106_upd_loc.ttcn:325 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:22.990528 TD1106_upd_loc.ttcn:326 Stop timer tl_guard2: 200 s
```

//Aquí terminan los comandos de preparación del HLR incluídas en la función f_Manual_Preparations();

//Intercambio de mensajes por el puerto TCAP

```
2013/Sep/28 18:21:22.998480 MapFunctions_Unidirectional.ttcn:90 Sent on TCAP_HLR to
system @TCAPasp_Types.ASP_TCAP_INVOKEreq : { dialogue_ID := 1, class :=
succAndFailReported (1), invoke_ID := 1, linked_ID := omit, operation := { local := 2 },
parameter := '301C040844444411111111F18107919471430100F00407919471430100F0'O, timeoutvalue
:= 70 }
2013/Sep/28 18:21:22.998600 MapFunctions_Unidirectional.ttcn:91 MAP: INVOKE sent:
{ dialogue_ID := 1, class := succAndFailReported (1), invoke_ID := 1, linked_ID := omit,
operation := { local := 2 }, parameter :=
'301C040844444411111111F18107919471430100F00407919471430100F0'O, timeoutvalue := 70 }
2013/Sep/28 18:21:22.998784 MapFunctions_Unidirectional.ttcn:94 Sent on TCAP_HLR to system
@TCAPasp_Types.ASP_TCAP_BEGINreq : { quality_of_Service := omit, destination_address := {
addressIndicator := { pointCodeIndic := '1'B, ssnIndicator := '1'B, globalTitleIndic :=
'0100'B, routingIndicator := '0'B }, signPointCode := '00000100101100'B, subsystemNumber
:= 6, globalTitle := { gti0100 := { translationType := '00'O, encodingScheme := '0001'B,
numberingPlan := '0001'B, natureOfAddress := '0000100'B, globalTitleAddress :=
'49172401111'H } } }, application_context_name := omit, originating_address := {
addressIndicator := { pointCodeIndic := '1'B, ssnIndicator := '1'B, globalTitleIndic :=
'0100'B, routingIndicator := '0'B }, signPointCode := '00000110011010'B, subsystemNumber
:= 7, globalTitle := { gti0100 := { translationType := '00'O, encodingScheme := '0001'B,
numberingPlan := '0001'B, natureOfAddress := '0000100'B, globalTitleAddress :=
'49173410000'H } } }, dialogue_ID := 1, user_information := omit }
2013/Sep/28 18:21:22.999035 MapFunctions_Unidirectional.ttcn:95 MAP: BEGIN sent:
{ quality_of_Service := omit, destination_address := { addressIndicator := {
pointCodeIndic := '1'B, ssnIndicator := '1'B, globalTitleIndic := '0100'B,
routingIndicator := '0'B }, signPointCode := '00000100101100'B, subsystemNumber := 6,
globalTitle := { gti0100 := { translationType := '00'O, encodingScheme := '0001'B,
numberingPlan := '0001'B, natureOfAddress := '0000100'B, globalTitleAddress :=
'49172401111'H } } }, application_context_name := omit, originating_address := {
addressIndicator := { pointCodeIndic := '1'B, ssnIndicator := '1'B, globalTitleIndic :=
'0100'B, routingIndicator := '0'B }, signPointCode := '00000110011010'B, subsystemNumber
:= 7, globalTitle := { gti0100 := { translationType := '00'O, encodingScheme := '0001'B,
numberingPlan := '0001'B, natureOfAddress := '0000100'B, globalTitleAddress :=
'49173410000'H } } }, dialogue_ID := 1, user_information := omit }
2013/Sep/28 18:21:22.999187 MapFunctions_Unidirectional.ttcn:96 setverdict(pass): pass ->
pass, component reason not changed
2013/Sep/28 18:21:22.999241 MapFunctions_Unidirectional.ttcn:1323 Start timer t_timer: 90
s
2013/Sep/28 18:21:23.027711 MapFunctions_Unidirectional.ttcn:1326 Message enqueued on
TCAP_HLR from system @TCAPasp_Types.ASP_TCAP_ENDind : { quality_of_Service :=
{ returnOptionRequired := true, sequencingInformation := true }, dialogue_ID := 1,
application_context_name := omit, components_present := true, user_information := omit }
id 1
2013/Sep/28 18:21:23.027794 MapFunctions_Unidirectional.ttcn:1328 Matching on port
TCAP_HLR succeeded:  matched
2013/Sep/28 18:21:23.027820 MapFunctions_Unidirectional.ttcn:1328 Receive operation on
port TCAP_HLR succeeded, message from system(): @TCAPasp_Types.ASP_TCAP_ENDind : {
quality_of_Service := { returnOptionRequired := true, sequencingInformation := true },
dialogue_ID := 1, application_context_name := omit, components_present := true,
user_information := omit } id 1
2013/Sep/28 18:21:23.027852 MapFunctions_Unidirectional.ttcn:1328 Message with id 1 was
extracted from the queue of TCAP_HLR.
2013/Sep/28 18:21:23.027868 MapFunctions_Unidirectional.ttcn:1330 MAP: END received:
{ quality_of_Service := { returnOptionRequired := true, sequencingInformation := true },
dialogue_ID := 1, application_context_name := omit, components_present := true,
user_information := omit }
2013/Sep/28 18:21:23.027895 MapFunctions_Unidirectional.ttcn:1331 setverdict(pass): pass -
> pass, component reason not changed
2013/Sep/28 18:21:23.028033 MapFunctions_Unidirectional.ttcn:1333 Message enqueued on
TCAP_HLR from system @TCAPasp_Types.ASP_TCAP_U_ERRORind : { dialogue_ID := 1, invoke_ID :=
1, errorvalue := { local := 1 }, parameter := omit, last_component := true } id 2
2013/Sep/28 18:21:23.028096 MapFunctions_Unidirectional.ttcn:1335 Matching on port
TCAP_HLR succeeded:  matched
2013/Sep/28 18:21:23.028120 MapFunctions_Unidirectional.ttcn:1335 Receive operation on
port TCAP_HLR succeeded, message from system(): @TCAPasp_Types.ASP_TCAP_U_ERRORind : {
```

```
dialogue_ID := 1, invoke_ID := 1, errorvalue := { local := 1 }, parameter := omit,
last_component := true } id 2
2013/Sep/28 18:21:23.028137 MapFunctions_Unidirectional.ttcn:1335 Message with id 2 was
extracted from the queue of TCAP_HLR.
2013/Sep/28 18:21:23.028152 MapFunctions_Unidirectional.ttcn:1337 MAP: ERROR received:
{ dialogue_ID := 1, invoke_ID := 1, errorvalue := { local := 1 }, parameter := omit,
last_component := true }
2013/Sep/28 18:21:23.028180 MapFunctions_Unidirectional.ttcn:1338 setverdict(pass): pass -
> pass, component reason not changed
2013/Sep/28 18:21:23.028445 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"SAAII:BLOCK=HISDAP3,SAE=500,NI=10000;"
2013/Sep/28 18:21:23.028498 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:23.028534 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "SAAII", commandParameters := "BLOCK=HISDAP3,SAE=500,NI=10000" } } }
2013/Sep/28 18:21:23.085061 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := ORDERED (1), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 11
2013/Sep/28 18:21:23.085126 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:23.085149 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := ORDERED (1),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 11
2013/Sep/28 18:21:23.085170 MML_Functions.ttcn:994 Message with id 11 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:23.085186 MML_Functions.ttcn:998 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:23.085202 MML_Functions.ttcn:1001 MML: Command OK, printout: ORDERED (1)
2013/Sep/28 18:21:23.085220 TD1106_upd_loc.ttcn:446 Start timer tl_guard1: 200 s
2013/Sep/28 18:21:23.708460 TD1106_upd_loc.ttcn:447 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0), parameters := { printout
:= { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES RESULT", "", "SAE    BLOCK
NI          NIE         NIR", " 500   HISDAP3       10000", "END" } } } } id 12
2013/Sep/28 18:21:23.709124 TD1106_upd_loc.ttcn:449 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:23.709161 TD1106_upd_loc.ttcn:449 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0),
parameters := { printout := { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES
RESULT", "", "SAE    BLOCK    NI          NIE         NIR", " 500   HISDAP3       10000",
"END" } } } } id 12
2013/Sep/28 18:21:23.709184 TD1106_upd_loc.ttcn:449 Message with id 12 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:23.709232 TD1106_upd_loc.ttcn:451 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:23.709250 TD1106_upd_loc.ttcn:452 Stop timer tl_guard1: 200 s
2013/Sep/28 18:21:23.709514 MML_Functions.ttcn:570 MML: Command will be sent to HLR:
"SAAII:BLOCK=HDSDAP3,SAE=500,NI=10000;"
2013/Sep/28 18:21:23.709568 MML_Functions.ttcn:986 Start timer t_timer: 15 s
2013/Sep/28 18:21:23.709601 MML_Functions.ttcn:990 Sent on MML_MTC_PORT to system
@MMLasp_Types.ASP_MML_REQUEST : { id := COMMAND (0), parameters := { command := {
commandName := "SAAII", commandParameters := "BLOCK=HDSDAP3,SAE=500,NI=10000" } } }
2013/Sep/28 18:21:23.796933 MML_Functions.ttcn:992 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PROCEDURE_PRINTOUT (1), parameters := {
procedurePrintout := { result := ORDERED (1), textbox1 := omit, faultCode := omit,
textbox2 := omit } } } id 13
2013/Sep/28 18:21:23.796997 MML_Functions.ttcn:994 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:23.797020 MML_Functions.ttcn:994 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id :=
PROCEDURE_PRINTOUT (1), parameters := { procedurePrintout := { result := ORDERED (1),
textbox1 := omit, faultCode := omit, textbox2 := omit } } } id 13
2013/Sep/28 18:21:23.797046 MML_Functions.ttcn:994 Message with id 13 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:23.797062 MML_Functions.ttcn:998 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:23.797078 MML_Functions.ttcn:1001 MML: Command OK, printout: ORDERED (1)
2013/Sep/28 18:21:23.797096 TD1106_upd_loc.ttcn:468 Start timer tl_guard1: 200 s
2013/Sep/28 18:21:24.264436 TD1106_upd_loc.ttcn:469 Message enqueued on MML_MTC_PORT from
system @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0), parameters := { printout
:= { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES RESULT", "", "SAE    BLOCK
NI          NIE         NIR", " 500   HDSDAP3       10000", "END" } } } } id 14
2013/Sep/28 18:21:24.265074 TD1106_upd_loc.ttcn:471 Matching on port MML_MTC_PORT
succeeded:  matched
2013/Sep/28 18:21:24.265112 TD1106_upd_loc.ttcn:471 Receive operation on port MML_MTC_PORT
succeeded, message from system(): @MMLasp_Types.ASP_MML_INDICATION : { id := PRINTOUT (0),
```

```
parameters := { printout := { answerPrintoutRecord := { "SIZE ALTERATION OF DATA FILES
RESULT", "", "SAE    BLOCK    NI          NIE         NIR", " 500   HDSDAP3       10000",
"END" } } } } id 14
2013/Sep/28 18:21:24.265134 TD1106_upd_loc.ttcn:471 Message with id 14 was extracted from
the queue of MML_MTC_PORT.
2013/Sep/28 18:21:24.265181 TD1106_upd_loc.ttcn:473 setverdict(pass): pass -> pass,
component reason not changed
2013/Sep/28 18:21:24.265200 TD1106_upd_loc.ttcn:474 Stop timer tl_guard1: 200 s
2013/Sep/28 18:21:24.265221 TD1106_upd_loc.ttcn:631 Unmapping port VLR_1_1106(3):TCAP_HLR
from system:TCAP_HLR.
2013/Sep/28 18:21:24.774394 TD1106_upd_loc.ttcn:631 Port TCAP_HLR was unmapped from
system:TCAP_HLR.
2013/Sep/28 18:21:24.774495 TD1106_upd_loc.ttcn:631 Unmap operation of
VLR_1_1106(3):TCAP_HLR from system:TCAP_HLR finished.
2013/Sep/28 18:21:24.774520 TD1106_upd_loc.ttcn:632 Unmapping port
VLR_1_1106(3):MML_MTC_PORT from system:MML_MTC_PORT.
2013/Sep/28 18:21:24.774600 TD1106_upd_loc.ttcn:632 Port MML_MTC_PORT was unmapped from
system:MML_MTC_PORT.
2013/Sep/28 18:21:24.774677 TD1106_upd_loc.ttcn:632 Unmap operation of
VLR_1_1106(3):MML_MTC_PORT from system:MML_MTC_PORT finished.
2013/Sep/28 18:21:24.774879 - Function f_Caso_1_Update_Location finished. PTC terminates.
2013/Sep/28 18:21:24.774894 - Terminating component type
ComponentDefinitions.TCAP_MML_Component.
2013/Sep/28 18:21:24.774911 - Port LDAP_OpenLDAP was stopped.
2013/Sep/28 18:21:24.774925 - Port LDAP_HLR was stopped.
2013/Sep/28 18:21:24.774942 - Port SoapLibraryHTTP was stopped.
2013/Sep/28 18:21:24.774954 - Port TCAP_HLR was stopped.
2013/Sep/28 18:21:24.774966 - Port MML_MTC_PORT was stopped.
2013/Sep/28 18:21:24.774977 - Port MML_MTC_PORT2 was stopped.
2013/Sep/28 18:21:24.774988 - Port MML_MTC_PORT3 was stopped.
2013/Sep/28 18:21:24.774999 - Port MML_MTC_PORT4 was stopped.
2013/Sep/28 18:21:24.775010 - Port MML_MTC_PORT5 was stopped.
2013/Sep/28 18:21:24.775023 - Port MML_MTC_PORT6 was stopped.
2013/Sep/28 18:21:24.775035 - Port MML_MTC_PORT7 was stopped.
2013/Sep/28 18:21:24.775046 - Port MML_MTC_PORT8 was stopped.
2013/Sep/28 18:21:24.775316 - Component type ComponentDefinitions.TCAP_MML_Component was
shut down inside testcase Caso_1_Update_Location.
2013/Sep/28 18:21:24.775341 - Final verdict of PTC: pass
2013/Sep/28 18:21:24.775406 - Disconnected from MC.
2013/Sep/28 18:21:24.775421 - TTCN-3 Parallel Test Component finished.
```

ANEXO II: PRESUPUESTO

En lo que sigue se incluye la planificación del proyecto con las tareas y el tiempo dedicado, así como el presupuesto.

Estas son las fases del proyecto y su consumo de recursos medido en hombre mes:

Fase	Hombre mes
Estudio TTCN-3	1
Estudio HLR Clásico	0,5
Estudio HLR FE	0,5
Estudio HLR Blade	0,5
Estudio Agile y Procesos	0,5
Estudio Update Location	0,5
Codificación Pruebas Update Location	0,75
Ejecución y Depuración Pruebas Update Location	0,75
Estudio Update GPRS Location	0,5
Codificación Pruebas Update GPRS Location	0,75
Ejecución y Depuración Pruebas Update GPRS Location	0,75
Estudio ACR	0,5
Codificación Pruebas ACR	0,75
Ejecución y Depuración Pruebas ACR	0,75
Estudio Cambio de IMSI	0,5
Codificación Pruebas Cambio de IMSI	0,75
Ejecución y Depuración Pruebas Cambio de IMSI	0,75
Generación documentación	1
TOTAL	**12**

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior
PRESUPUESTO DEL PROYECTO

1. **Autor:**
   Ignacio García Medina
2. **Departamento:**
   Ingeniería Telemática
3. **Descripción del Proyecto:**
   Titulo: Automatización de pruebas funcionales del HLR usando TTCN-3
   Duración: 12 meses
   Tasa de costes indirectos: 20%
4. **Presupuesto total del proyecto:**
   40.828 €
5. **Desglose presupuestario (costes directos):**

PERSONAL				
Apellidos y nombre	**Categoría**	**Dedicación[1] (hombres/mes)**	**Coste (hombre/mes)**	**Coste (€)**
García Medina, Ignacio	Ingeniero	12	2.690,63	32.287,50
Soto Campos, Ignacio	Tutor	0.18	4.300,00	774,00
Gómez Ruíz, Jesús	Tutor	0,18	4.300,00	774,00
			TOTAL	**33.835,50**

[1] 1 hombre/mes = 131,25 horas. Máximo anual de dedicación de 12hombres/mes (1575 horas).
Máximo anual para DPI de la Universidad Carlos III de Madrid de 8,8 hombres/mes (1.155 horas).

EQUIPOS					
Descripción	**Coste (Euro)**	**% Uso dedicado al proyecto**	**Dedicación (meses)**	**Periodo de depreciación**	**Coste [2]imputable**
Servidor para ejecución de Máquina Virtual Linux SUSE 10	180,00	100	12	60	36,00
Servidor para ejecución herramienta SEA	360,00	100	12		72,00
Licencia pila EINSS7					80,00
				TOTAL	**188,00**

6. Resumen de costes:

PRESUPUESTO COSTES TOTALES	
Personal	33.836
Amortización	188
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes indirectos	6.805
TOTAL	**40.828**

[2] Fórmula de cálculo de la Amortización:

$\frac{A}{B} xCxD$ A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

Referencias

[i] *Historia del teléfono móvil (Wikipedia.)* Disponible [Internet]: http://es.wikipedia.org/wiki/Historia_del_tel%C3%A9fono_m%C3%B3vil

[ii] *Telefonía móvil en España (Wikipedia)*. Disponible [Internet]: http://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil_en_Espa%C3%B1a [20/02/2013]

[iii] Unión Internacional de Telecomunicaciones. *Informe Medición de la Sociedad de la Información*. Disponible [Internet]: http://www.itu.int/dms_pub/itu-d/opb/ind/D-IND-ICTOI-2012-SUM-PDF-S.pdf [5/03/2013]

[iv] Ken Schwaber and Jeff Sutherland. Octubre 2011. *The Scrum Guide*. Scrum.org.

[v] *Protocolo LDAP (Wikipedia)*. Disponible [Internet]: http://es.wikipedia.org/wiki/LDAP [20/03/2013]

[vi] *TTCN-3 Standard Overview (Testing Technologies).* Disponible [Internet]: http://www.testingtech.com/ttcn3/standard_overview.php

[vii] Alex Rennoch, Claude Desroches, Theo Vassiliou and Ina Schieferdecker. *TTCN-3 Quick Reference Card V0.47*. Disponible [Internet]: http://www.blukaktus.com/TTCN3QRC_viewme.pdf

[viii] ETSI - Estándar TTCN3. *ETSI ES 201 873-1 V4.1.1 (2009-06)*. Disponible [Internet]: http://www.etsi.org/deliver/etsi_es/201800_201899/20187301/04.01.01_60/es_20187301v040101p.pdf

[ix] Especificación de la funcionalidad *LOCATION UPDATING IN HLR*. Número de documento: 6/155 17-ANT 238 01 Uen N

[x] Especificación de la funcionalidad *GPRS LOCATION UPDATING IN HLR.* Número de documento: 188/155 17-ANT 238 01 Uen D

[xi] Especificación de la funcionalidad *IMSI CHANGEOVER IN HLR.* Número de documento: 44/155 17-ANT 238 01 Uen E

[xii] Especificación de la funcionalidad *HANDLING OF ANONYMOUS CALL REJECTION IN HLR.* Número de documento: 241/155 17-ANT 238 01 Uen A